செம்மொழியில் கற்போம்

ஷெல் ஸ்கிரிப்ட்

shell script

பணியா. பிரசன்னா

செம்மொழியில் கற்போம்

ஷெல் ஸ்கிரிப்ட்

பணியா. பிரசன்னா

askprasanna@gmail

மின்னூல் வெளியீடு :

FreeTamilEbooks.com

அட்டைப்படம், மின்னூலாக்கம் :

மீ .வேல். பிரசன்னா

udpmprasanna@gmail.com

உரிமை:

Creative Commons Attribution - ShareAlike 4.0 International License.

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -3- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

இந்நூல்
பெற்று, வளர்த்து, தவறிடும் போது தடுத்தாட்கொண்டு அறநெறியில் பேணிக்காத்திடும் எனது அம்மாவிற்குக் காணிக்கை.

சி.மரிய மதலேனா பி.ஏ.பி.எட்., இடைநிலை ஓய்வு ஆசிரியை

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -4- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

பொருளடக்கம்

நூலுரை:..11 இளவலுரை:...12 எழிலுரை:...13 நன்றிகள்:...14 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - பகுதி 1..................15 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – பகுதி 2 தரவு வகைகள்.21 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 3 செயல் வகைகள், நிலைகள்..27 செம்மொழியில்கற்போம்ஷெல்ஸ்கிரிப்ட் – 4 இயங்குதளத்தின்ஏற்றநிலை, மாறிகளைப் பயன்படுத்துதல்.............33 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 5 இயங்குதளத்தின் ஏழு ஓடு நிலைகள்..38 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 6 செயற்பாடு அல்லது நிரல்துண்டு (Functions)..43 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் 7 - கட்டுப்பாட்டு அமைவுகள் (if condition)..46 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட்– 8 சுழற்சி அல்லது ஆகக்கட்டளை (for loop)..52 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட்– 9 சுழற்சி அல்லது ஆகக்கட்டளை (for loop) தொடர்ச்சி...57 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -10 வளைவுக் கட்டளையின் வேறு வகைகள்..62 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -11 பொழுதெலாம் கட்டளை அல்லது while loop – entry controlled loop...........................67

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -5- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -12 வரையிலும் கட்டளை அல்லது until loop – exit controlled loop..............................72 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -13. தேர்வுக் கட்டளை அல்லது தேர்வாணை (case statement).....................................77 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -14. தேர்வுக் கட்டளை இருந்தால் கட்டளை வேறுபாடுகள் (case statement vs else if ladder)..82 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -15. சுழற்சியில் முறிவுக்கட்டளை (break statement in loop).........................86 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -16. சுழற்சியில் தொடர் கட்டளை அல்லது இடைவிடாக் கட்டளை (continue statement in loop) ....91 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -17 கட்டளைவரி அளவுருக்கள்அல்லதுகட்டளைவரிதருமதிப்புக்கள் (command line parameters or command line arguments).........................96 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -18.கட்டளைவரி அளவுருக்கள் அல்லது கட்டளைவரி தருமதிப்புக்கள் (command line parameters or command line arguments) - தொடர்ச்சி......100 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -19. உறக்கக் கட்டளை அல்லது தூக்கக் கட்டளை (sleep command)...........................105 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -20. அளவுருக்கள் அல்லது தருமதிப்புக்கள் (command line arguments or command line parameters) தொடர்ச்சி...109 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -21........................114 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 22 மாற்றுக் கட்டளை (shift command)..119 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 23 Trap command (கண்ணி -கட்டளை)...124

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 24 (getopts command) பொருத்தம் பெறும் கட்டளை.................................129 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 25 (cut command) வெட்டுக் கட்டளை..134 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 26 (paste command) ஒட்டுக் கட்டளை...137 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 27 (tput command) இடு கட்டளை..140 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 28 (tput command) இடு கட்டளை..145 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 29 (tput command) இடு கட்டளை தொடர்ச்சி...150 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 30 (nohup command) செயலிழக்காக் கட்டளை..154 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 31.......................157 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 32.......................161 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 33.......................165 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 34 தப்பிடும் விசைகள் (escape sequences)...170 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 35 வெளியேறும் வழிக்கட்டளை (exit command)...176 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 36 உள்ளார்ந்த கட்டமைப்புக் கட்டளைகள் (Internal Commands and Builtins)180 செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 37 இரட்டை அடைப்புக் குறி...185 முடிவுரை:...189 நூல் ஆசிரியர்:...190

நூலுரை:

“செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட்” இந்தத் தலைப்பைச் சொல்லும் பொழுதே ஏதோ திருமொழியைச் சொல்கின்றோம் என மனதிற்குத் தோன்றுகிறது. இலினக்சு (Linux) இயங்குதளத்தில் ஏறத்தாழ பன்னிரெண்டு ஆண்டுகள் பட்டறிவு இருப்பினும், இப்பொழுதுதான் இது போன்ற ஒரு நூலைத் தமிழில் தரத் தருணம் பிறந்திருக்கிறது.
இந்நூலில் ஒவ்வொரு பகுதியிலும், அதில் அமைந்துள்ள ஆங்கிலக் கலைச்சொற்களுக்கு இணையாகத் தமிழ்ச்சொற்களைத் தந்திருக்கிறேன். செம்மொழியில் என்று நூல் தொடங்குவதால், ஒவ்வொரு பகுதியிலும் ஒரு கவிதை வரைந்திருக்கிறேன். தமிழ் வழிக்கல்வி கற்றோர் இதை பெரிதும் விரும்புவர். இந்தக் கவிதைகள் புதுக்கவிதையில் சேராமல், நான் என்றோ படித்த மரபுக் கவிதையின் தாக்கத்தில் விளைந்தவை. இவை முழுமையாக எந்தப் பா வகையிலும் சேராமல் இருப்பினும், மரபுக்கவிதை போன்றே அமைந்திருப்பது சிறப்பு. இலினக்சு ஏற்கனவே தெரிந்திருப்போர், இந்நூலைப் படிப்பின் ஷெல் ஸ்கிரிப்ட்டை எளிய தமிழில் கற்கலாம். நான் சொல்கிறேன், நீங்கள் கேளுங்கள் என்ற வகையில் நூல் இல்லாமல், கற்போம் என்ற பதத்தின் மூலம், நானும் உங்களுடன் சேர்ந்து கற்கிறேன் என்பதை அறியத்தருகிறேன்.
எனக்குத் தெரிந்தவைகளை இயன்ற வரையில் இனிய தமிழில் எடுத்துரைக்க முன்றிருக்கிறேன். ஒவ்வொரு பகுதியிலும் பல நிரல்கள் இடம் பெற்றிருக்கின்றன. அவற்றின் விளக்கங்கள் பத்திகளில் இடம் பெற்றுள்ளன. நிரல்+பா=நிரற்பா என்ற கிளவியின் மூலம் நானறிந்த தமிழை நாடறியச் செய்ய விழைகின்றேன்.
நூலில், முதலில் நிரற்பா வரும். பிறகு அதற்கான விளக்கம் உண்டு. நிரல்களுக்கு வரிசை எண்கள் தரப்பட்டுள்ளன. புரியாத சொற்களுக்கு அல்லது பொதுவாகப் பயன்படுத்தும் ஆங்கிலப் பதங்களுக்கு விளக்கங்கள் அடைப்புக்குறிகளுக்குள் வழங்கப்பட்டுள்ளன. பகுதிகளின் ஈற்றில் கலைச்சொற்களும் அவற்றிற்கான விளக்கங்களும் ஆங்கிலத்தில் விளக்கப்பட்டுள்ளன. படங்கள், அதற்கான விளக்கங்கள், அட்டவணைகள் ஆகியனவும் அளிக்கப்பட்டுள்ளன.

“கற்போம்” என்று தமிழ் கம்ப்யூட்டர் இதழில் தொடராக வெளிவந்த இந்தக் கட்டுரைகளைத் தொகுத்து ஒரு நூலாகத் தருவதில் பெருமகிழ்ச்சி கொள்கிறேன். எனக்கு இது மிகவும் பயனுள்ள நூலாகத் தெரிகின்றது. உங்களுக்கும் அதே உணர்வு ஏற்படும் என்றே கருதுகின்றேன்.

இளவலுரை:

உலகெங்கும் இருக்கின்ற அன்புத் தமிழ் ஆர்வலர்களுக்கு, எனது பணிவான வணக்கங்கள். எனது தமையனார் பணியா. பிரசன்னா அவர்களின் செம்மொழியில் கற்போம் ஷெல்ஸ்கிரிப்ட் என்ற இந்நூல் கணினி அறிவியலில் ஒரு மைல் கல். “நீரின்றி அமையாது உலகு” என்ற கூற்றைப் போல், கணினி, கைபேசி, மற்றும் எண்ணிலடங்கா மின் கருவிகள்(Devices /Gedgets) இயங்கத் தேவை ஒரு வலுவான இயங்குதளம் (Operting System).
அத்தகைய இயங்குதளங்களுள் Linux முடிசூடா மா மன்னன். Linux இயங்குதளத்தில் பணியா.பிரசன்னா மின்னூலாசிரியரின், திறன் என்பது நான் அருகில் இருந்து கண்டு வியந்தது. அவர் தமிழின் பால் கொண்டுள்ள அவா தமிழ் கம்ப்யூட்டர் இதழின் வாயிலாக தமிழகமெங்கும் விரிந்து பரவலாயிற்று. பின்னாளில், இத்தகு மின்னூலின் வாயிலாக உலகத் தமிழர்களின் (குறிப்பாக கனடா, நார்வே,ஃபிரான்ஸ், மற்றும் ஈழம்) அன்பைப் பெற்று தொடர்ந்து வீர நடை போடுகிறது. அவர் மிகச்சிறப்பாக ஆங்காங்கே சிறு நிரற்பாக்களையும் அதன் விளக்கத்தையும் கூறி, படிப்பவரை ஆர்வமடையச்

செய்திருக்கிறார். கணினி அறிவியலை தமிழின் வழியாக மென்மேலும் சிறப்பு பெறச் உலகத் தமிழர்களின் ஆதரவு பெருகிட வாழ்த்துகிறேன்.
அன்புத்தம்பி,
பா.அறிவு ஆரோக்கிய இராஜேஷ்.
மென்பொருள் கட்டுமான மேலாளர், டிசிஎஸ், பெங்களூரு.

எழிலுரை:

நாம் எங்கு செல்கிறோம் என்று நினைத்துப் பார்க்க முடியாத, அளவிற்கு தொழிற்நுட்பமானது வெகு விரைவாக வளர்ந்து வருகிறது. உலக மயமாக்கலின் வாயிலாக, நாம் பல்வேறு வாய்ப்புகளைப் பெற்று நாள் தோறும் வளர்ந்து வருகிறோம். வெளிநாட்டுக்குரிய பள்ளிகளும் இந்தியாவில் அதிகரித்து வருகிறமையால் அனைத்து வெளிநாட்டு வாய்ப்புகளும் எளிமையாய் நம் நாட்டிற்கு வந்து விடுகின்றன. அதைப்போலவே, குழந்தைகளும் தங்கள் தாய்மொழி தவிர்த்து பிறமொழிகளைப் பயில்வதில் ஆர்வம் காட்டி வருகிறார்கள். ஆராய்ச்சி என்பது பிறமொழிகளைப் பயின்று அதன் மூலம் வருவதில்லை. தாய்மொழியில் பயின்று அதைத் திறம்படக் கற்று அதிலிருந்து ஆய்வறிக்கை வருவதே சிறந்தது.
தாய்மொழி வழிக்கற்றலே அதன் வழி தடையறச் சிந்திக்கவும் பெரிதும் உதவுகிறது. இன்றைய நாளில் எந்தப் பணி செய்தாலும், அதற்கு கணினி கல்வியறிவு தேவைப்படுகிறது. கல்வி கற்றவர்கள் தங்கள் தாய்மொழியில் பயில இந்நூல் பெரிதும் உதவும் என்பதில் எள் முனையளவும் ஐயமில்லை.

ஆசிரியரின் அளப்பரிய இப்பணியைப் பாராட்டுகிறேன். தமிழ் வழிக்கல்வி கற்றோர் இந்நூல் பயின்று இன்புற அன்புடன் அழைக்கிறேன்.

அன்பு மனைவி,

விர்ஜின் பிரசன்னா.

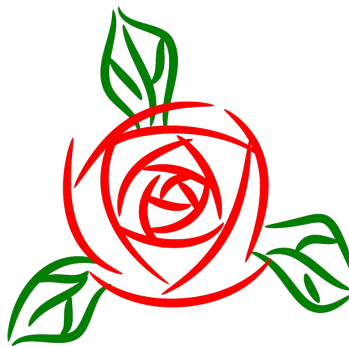
திட்ட ஒருங்கிணைப்பாளர்,

செயிண்ட் ஜான்ஸ் கல்லூரி, பெங்களூரு

நன்றிகள்:

எல்லாம் வல்ல இறைவன் தொடங்கி, இளவலுரை தந்த எனது தம்பிக்கும், எழிலுரை தந்த எனது மனைவிக்கும், மின்னுல் எழிலுற அமைத்த எனது அருமை உடன்பிறப்பு மீ.வேல்.பிரசன்னாவுக்கும், மற்றும் தமிழ் கம்ப்யூட்டர் பதிப்பகத்தாருக்கும் மற்றும் இந்நூல் வெளிவரத் தெரிந்தோ, தெரியாமலோ உதவியோர் அனைவருக்கும் எனது நெஞ்சார்ந்த நன்றிகள்.

“உளன் எனின் உளன் என்பதும்
இலன் எனின் இலன் என்பதும்
உலகோர் கூற்றே; அன்பே
ஆண்டவன் என்பது அறத்தின் வாழ்த்தே!!”



செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - பகுதி 1

“பயிற்சியால் வருமாம் பயனர் தோழமை
கட்டளை வரிசை கொண்டி யங்கும்
தானாய்ச் செயல்கள் தடையறச் செய்யும் ஃபின்லாந் தந்தமென் பொறிக்குறு நிரலே” நிரற்பா-1

முன்னுரை:

இயங்குதளம் என்பது பயனருக்கும், கணினிக்குமிடையே இருக்கும் ஒரு இடைமுகப்பாகும். பொதுவாக ஒரு இயங்குதளத்தினை நிறுவல் செய்வது மட்டுமே பயனரின் அனைத்துத் தேவைகளையும் பூர்த்தி செய்து விடாது. பயனரின் அன்றாடத் தேவைகள் தானியங்கு மயமாக்கப்படல் வேண்டும். அப்பொழுதான் பொறிநிறைஞரின் வேலை எளிதாகும். பயனரின் தேவைகள் விரைவாக முடியும். அவ்வாறு தானியங்கு மயமாக்கலுக்கு, இயங்குதளத்தில் உள்ள கட்டளைகளை தொடர்பு படுத்தி நிரல்கள், குறுநிரல்கள் எழுதப்படல் வேண்டும். அவையும் பொதுமயமாக்கலுக்கு ஏற்ற வண்ணம் எழுதப்படல் வேண்டும். அதற்காக நாம் ஷெல் ஸ்கிரிப்ட் கற்க வேண்டும்.

ஷெல் ஸ்கிரிப்ட் என்றால் என்ன?

இலினெக்ஸு இயங்குதளத்தில் பல்லாயிரக்கணக்கான கட்டளைகள் உள்ளன. அவற்றை நமது செயலுக்கு ஏற்ற வண்ணம் எடுத்து குறுநிரல்கள் எழுதப்படுவதே ஷெல் ஸ்கிரிப்ட் என்றழைக்கப்படுகிறது. ஷெல் ஸ்கிரிப்ட் என்பது சி, சி++ போன்று ஓர் உயர்நிலை மொழி கிடையாது. தன்மயமாக்கலுக்காகவே ஷெல்ஸ்கிரிப்ட்டுகள் பயன்படுத்தப் படுகின்றன. கணினி மொழிகளைப் பொதுவாக இரண்டுவகையாகப் பிரிக்கலாம். ஒன்று கம்பைலர் முறை மொழிகள் இரண்டு இன்டர்ப்ரெட்டர் மொழிகள். கம்பைலர் மொழிகள் பிழைகளை மொத்தமாகக் காண்பிக்கும். இன்டர்பிரட்டர் மொழிகள் ஒவ்வொரு படியாகக் (step by step execution) காண்பிக்கும். இதில் ஷெல்ஸ்கிரிப்ட் என்பது இரண்டாம் வகையைச் சேர்ந்தது. ஷெல் ஸ்கிரிப்ட்டுகள் பொதுவாக எல்லாவகையான இலினக்ஸீ இயங்குதளங்களிலும்

இயங்கும். விண்டோஸ் வகை இயங்குதளங்களில் இதற்கு இணையாக பேட்ச் ஸ்கிரிப்டுகள் பயன்படுத்தப்படுகின்றன.

யாரெல்லாம் படிக்கலாம்:

பொறி நிறைஞர்கள் (System Administrators), மென்பொறிஞர்கள் (software engineers), நிரலர்கள் (programmers), யுனிக்ஸ், லினக்ஸ் ஆர்வலர்கள் (Unix & Linux enthusiasts) ஆகியோருக்குப் படித்து எளிதில் புரிந்து கொள்ளும் வண்ணம் இத்தொடர் அமைக்கப் பெற்றிருக்கிறது. தொடரைப் படித்துப் புரிந்து கொள்ள இயங்குதள அமைப்பு (OS architecture), குறைந்தபட்சம் தொடக்க நிலை நிரலெழுதும் திறமை (at least beginner level programming skills) ஆகியவை இருக்க வேண்டும்.

ஷிபேங்க் (Shebang)

ஷிபேங் என்பது நாம் எழுதக்கூடிய குறுநிரல், எந்த வகையான ஷெல்லில் இயக்கப்பட வேண்டும் என்பதனைக் குறிக்கிறது. இது ஷிபேங் ஷாபேங் ஹாஷ்பேங், பௌண்ட் பேங், ஹாஷ் எக்ஷ்கிலெம், ஹேஷ் பிலிங் என்று பலவாறாக அழைக்கப்படுகிறது. இதை #! என்று குறுநிரலில் குறிப்பிடுவது வழக்கம்.

ஷெல்லின் வகைகள்:

இலினெக்ஸைப் பொறுத்தவரையில் பலவகையான ஷெல்கள் காணப்படுகின்றன. ஆனால் நமது குறுநிரல்கள் குறிப்பிட்ட ஷெல்லில் மட்டுமே நடைபெறவேண்டுமென்பதற்காகவே நாம் ஷிபேங் என்று நிரலின் தொடக்கத்திலேயே குறிப்பிட வேண்டும். உள்ளிருப்பாக ஷெல் ஸ்கிரிப்ட்டுகள் பேஷ் ஷெல் எனப்படும் ஷெல்லிலேயே இயக்கப்படும். ஆனால் இது ஒவ்வொரு பொறி பொறுத்தும் மாறுபடும். எனவேதான் நாம் இதனை ஒவ்வொரு நிரலின் தொடக்கத்திலும் செய்ய வேண்டும். இல்லையேல் ஒரு பொறியில் சிறப்பாக இயங்கும் குறுநிரல் வேறு பொறியில் (உள்ளிருப்பாக இருக்கும் ஷெல் மாற்றப்பட்டிருந்தால்) சரிவர இயங்காது.
1. பார்ன் ஷெல் (Bourne shell (sh))

2. பார்ன் எகெயின் ஷெல் (Boune Again SHell (BASH))

3. சி ஷெல் (C Shell (csh))

4. [illegible] (TC Shell (tsh))

5. [illegible] (Korn shell (ksh))

[illegible], [illegible]. **sh** [illegible], [illegible]. [illegible].

[illegible]:

[illegible]. [illegible] **zsh** [illegible]. [illegible].

Criteria		sh	ksh	bash	zsh	csh	tcsh
Configurability	1	-	+	++	+++	+	++
Execution of commands	2	+	+	+	++	+	++
Completion	3	- -	+	++	+++	+	++
Line editing	4	-	+	++	++	-	++
Name substitution	5	+	+	++	++	+	++
History	6	- -	+	++	++	+	++
Redirections and pipes	7	+	+	+	++	+	+
Spelling correction	8	- -	- -	- -	+	- -	+
Prompt settings	9	+	+	+	++	+	++
Job control	10	- -	+	+	+	+	+
Execution control	11	+	+	+	+	+	+
Signal Handling	12	+	+	+	+	-	-

+++ [illegible]

++ [illegible]

+ [illegible]

- [illegible].

-- [illegible]

[illegible], [illegible], [illegible].

முன்னறிவுத் தேவைகள்:

ஷெல் ஸ்கிரிப்ட் மொழி பயில்வதற்கு சி மொழி தெரிந்திருக்க வேண்டும் என்று சிலர் சொல்வதுண்டு. கண்டிப்பாக தெரிந்திருக்க வேண்டும் என்பதல்ல. தெரிந்திருந்தால் ஷெல் ஸ்கிரிப்ட் பயில்வதற்கு எளிதாக இருக்கும். லினக்சு இயங்குதளத்தின் கட்டளைகள் நன்கு தெரிந்திருக்க வேண்டும். அப்பொழுது எல்லா வகையான செயல்களையும் நாம் நன்கு கட்டுப்படுத்தி, தானியங்கு நிரல்களை எழுத முடியும் . எனவே ஷெல் ஸ்கிரிப்ட் என்பது பொதுவாக சில லினக்சு கட்டளை வரிகளின் தொடர்ச்சியாகும். புதிதாக ஷெல் ஸ்கிரிப்ட் கற்றுக் கொள்வதற்கும், ஏற்கனவே கற்றவர்கள் அதை மேம்படுத்திக் கொள்வதற்கும் இத்தொடர் துணைபுரியும்.

நிரல் **1:**

```
#!/bin/bash
echo “Welcome to Linux Shell script world”
```

நிரல் இயக்கும் முறை:

மேற்கூறிய நிரலை ஏதேனும் ஒரு உரைத் தொகுப்பானில் தட்டச்சு செய்து, அல்லது வி(ம்) தொகுப்பானில் **(vi(m) editor)** எழுதி சேமித்துக் கொள்ளவும். எடுத்துக்காட்டாக , **script1.sh** என பெயரிடலாம். குறுநிரலின் கோப்பானது, **.sh** என்ற பின்னொட்டுடன் இருக்க வேண்டிய தேவையில்லை. இருப்பினும், இந்தப் பின்னொட்டு நிரலர்களுக்கு குறுநிரல்களைப் பிரித்தறிய உதவுகிறது.

#sh -x script1.sh என்ற கட்டளைவரி குறிப்பிட்ட குறுநிரலை வரிவரியாக இயக்கி வெளியீட்டைத் தருகிறது.

#sh -v script1.sh என்ற வரியானது வெளியீட்டை நிரலருக்குப் புரியும் வண்ணம் விளக்குகிறது.

இருப்பினும் இக்கட்டளை வரிகளாவை ஒரு குறுநிரலை பிழைதிருத்தப் பயன்படுத்துதல் சிறப்பு . ஏற்கனவே பிழைதிருத்திய நிரல்களை கீழ்க்காணும் கட்டளைவரிகள் மூலமாக இயக்கிப்பார்க்கலாம்.

#chmod +x script.sh இந்தக்கட்டளை வரி ஒரு நிரலை அனைத்துப் பயனர்களும் இயக்கிப்பார்க்கும் உத்தரவினைத் தருகிறது. இவ்வரி தராவிட்டால் நிரலை பயனர்கள் யாரும் இயக்க முடியாது.

#./script.sh இது ஒரு நிரலை இயக்கி வெளியீட்டைத் தருகிறது. பிழைகள் இருப்பின், பிழைகளைப் படிப்படியாகக் காண்பிக்கும்.
இங்கு படிப்படியாக என்பது, நான்கு வரி நிரலில் இரண்டாம் வரியிலும் மூன்றாம் வரியிலும் பிழையிருப்பின், முதலில் இரண்டாம் வரியினையும், அது சரி செய்யப்பட்ட பின்பே மூன்றாம் வரியின் பிழையையும் காண்பிக்கும். மேற்குறிப்பிட்ட நிரல் Welcome to Linux Shell script world என்ற வெளியீட்டினைத் தரும். echo கட்டளை தன்னுள் இருக்கும் வாக்கியத்தினை வெளிக்காட்டுகிறது.
ஐயம்: ஷெல் ஸ்கிரிப்ட் விண்டோஸ் இயங்குதளத்தில் இயங்குமா?

தீர்வு: பொதுவாக லினக்சில் இயங்கும் ஷெல் ஸ்கிரிப்ட்டுகள், விண்டோசில் இயங்காது. சில பொதுவான echo போன்ற கட்டளைவரிகள் ஒரே மாதிரியான வெளியீட்டைத் தரும். விண்டோசுக்கென்று தனியாக batch file programming இருக்கிறது. (filename.bat) அவற்றில் விண்டோசுக்கான கட்டளைகளை இயக்கிப் பார்த்து ஸ்கிரிப்டுகள் எழுத முடியும். அவை முற்றிலும் விண்டோஸ் இயங்குதளத்திற்கு மட்டுமே உரியன. விண்டோஸ் இயங்குதளத்தில் உள்ள பவர் ஷெல் மூலமாக அனைத்து வகையான விண்டோஸ் கட்டளைகளையும் இயக்கலாம். இதன் முதல் பதிப்பான பவர் ஷெல் 1 விண்டோஸ் எக்ஸ்பி எஸ்பி2 விலேயே வந்திருந்தாலும், விண்டோஸ் ஏழு, எட்டு இயங்குதளங்களில் இதன் மேம்பட்ட பதிப்பினைப்

[illegible]. [illegible]. **Cygwin** [illegible] **(Linux Simulation)** [illegible].

```
Windows PowerShell
PS C:\Users\jtwist> get-help about_regular_expression
TOPIC
    Regular expressions


SHORT DESCRIPTION
    Using regular expressions in Cmdlet parameters in the Windows PowerShell


LONG DESCRIPTION



    PowerShell supports a number of the regular expression characters:

        Format   Logic                              Example
        -------- ---------------------------------- -----------------------
        value    Matches exact characters           "book" -match "oo"
                 anywhere in the original value
        .        Matches any single character       "copy" -match "c..y"
        [value]  Matches at least one of the        "big" -match "b[iou]g"
                 characters in the brackets
        [range]  Matches at least one of the        "and" -match "[a-e]nd"
                 characters within the range.
                 The Use of a hyphen (-) allows
                 specification of contiguous
                 character.
        [^]      Matches any character except       "and" -match "[^brt]nd"
                 those in brackets
        ^        Matches the beginning              "book" -match "^bo"
                 characters
        $        Matches the end characters         "book" -match "ok$"
        *        Matches zero or more instances     "baggy" -match "g*"
                 of the preceding character
        ?        Matches zero or one instance       "baggy" -match "g?"
                 of the preceding character
        \        Matches the character that         "Try$" -match "Try\$"
                 follows as an escaped character


    PowerShell supports the character classes available in .NET regular
    expressions
        Format   Logic                              Example
        -------- ---------------------------------- -----------------------
        \p{name} Matches any character in the       "abcd defg" -match "\p{Ll}+"
                 named character class specified
```

[illegible].

i) [illegible]

ii) [illegible]

iii) [illegible]

iv) [illegible]

v) [illegible]

[illegible].

System Call Interface

Process Subsystem | Memory Subsystem | File Systems | Device Control | Network Subsystem

Concurrency | Virtual Memory | VFS | tty access | Connectivity

Arch code | Memory manager | FS types / Block devs | Character devices | Network layers / Network drivers

HARDWARE

□□□□□□□□□□□□□:

□□□□□□□□□□□□□□ – generalization

□□□□□□□□□□□ – interface

□□□□□□□□□□□□□ – scripts

□□□□□□□□□□□□□ – automation

□□□□□□□□□□□□ – by default

□□□□ – system

□□□□□□□□□ – architecture

(□□□□□□□□...)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – பகுதி 2 தரவு வகைகள்

எண்கள் உரைகள் என்றிரு தரவுகள் கொண்டு புனைந்திட கணக்கு எளிமை
ஏரணப் பிழையற் றெழுதிடல் வேண்டும்
எண்ணிய தீர்வு இன்பமாய் வரவே
நிரற்பா – 2

ஒரு இயங்குதளமானது பின்வரும் ஐந்து வகையான மேலாண்மைகளைச் சரிவரகையாள வேண்டும்.
vi) செயலி மேலாண்மை (process management)

செயலி மேலாண்மையானது ஓர் இயங்குதளத்திலுள்ள அனைத்து வகையான செயலிகளையும் கண்டுணர்ந்து அவற்றை உரிய வகையில் இயக்குதளைக் குறிக்கிறது. செயல் என்பது என்ன ஒரு வகையான வினையாகவும் இருக்கலாம். எடுத்துக்காட்டிற்கு கூட்டல், கழித்தல், வகுத்தல், பெருக்கல் என அடிப்படை கணக்கீட்டு வினைகளையும் எடுத்துக்கொள்ளலாம்.
vii) நினைவக மேலாண்மை (memory management)

நினைவக மேலாண்மையானது எந்த வகையான செயலுக்கு எவ்வளவு நினைவகம் தேவைப்படும் என்பதை ஆய்ந்தறிந்து செயற்பட வேண்டும். இதிலும் முதன்மை நினைவகம், இரண்டாம் நினைவகம் என இரண்டு வகைகள் உண்டு. முதன்மை நினைவகம் பொதுவாக RAM (Random Access Memory) ஐக் குறிக்கிறது. இரண்டாம் நினைவகம் வன்வட்டினைக் குறிக்கிறது.
மெய்நிகர் நினைவகம்: இது வன்வட்டின் ஒரு பகுதியாக இயங்கினாலும், பொதுவாக RAM ஆனது முழுவதும் செயலிகளால் கையகப்படுத்தப்பட்டால்,
இது RAM ஆகச் செயல்படும். இலினக்சு இயங்குதளத்தில் இது பொதுவாக ஒன்றரை முதல் இரண்டு மடங்கு RAM அளவினைக் கொண்டிருக்கும் வண்ணம் அமைக்கப்படல் வேண்டும். இது பொதுவாக விதியாக இருப்பினும், அதிக

அளவு RAM பயன்படுத்தப்படும் பொழுது இதற்கு விதிவிலக்குகளும் உள்ளன.
viii) கோப்பு மேலாண்மை (file management)

கோப்பு என்பது பல்வேறு வகையான தரவுகளின் அல்லது பதிவுகளின் கூட்டமைவு ஆகும். இவற்றைச் சரிவர மேலாண்மை செய்தல் இயங்குதளத்தில் வேலைகளில் இன்றியமையாததாகும். அவ்வாறு செய்யாவிடில் பயனர் கோப்புக்களை அணுகும் பொழுது குழப்பம் நேரலாம். இலினக்சானது ext4, ext3, ext2, vfat, swap கோப்பவமைவுகளை தன்னகத்தே கொண்டிருந்தாலும், அது VFS என்ற Virtual File System கொண்டே பொதுவாக பயனர்களுக்கு தனது வெளியீட்டினைத் தருகிறது. இங்கு VFS என்பது ஒரு தற்காலிக கோப்பமைவாக உள்ளது. ஒரு குறிப்பிட்ட கட்டளை எவ்வாறு இயங்குகிறது என்பதனை ஓர் எடுத்துக்காட்டின் மூலம் காணலாம்.
ls என்ற கட்டளையானது listing என்பதன் சுருக்கமாகும். அதாவது ஒரு கோப்பமைவில் அல்லது ஓர் அடைவில் இருக்கும் கோப்பு மற்றும் துணை அடைவுகளை காட்டும் கட்டளையாகும். இந்தக் கட்டளை கொடுக்கப்பட்டவுடன் அது ஒரு fork (துணைச்செயல்) ஐ உருவாக்கி, execute(“ls”) என ஏற்கனவே சி மொழியில் எழுதப்பட்ட ஒரு நிரலை அழைக்கும். (இலினக்சு சி மொழியில் எழுதப்பட்ட இயங்குதளம் என்பதறிக.) அது குறிப்பிட்ட கருவியினைத் திறந்து (வன்வட்டு/குறுவட்டு/நினைவக அட்டை) தற்காலிக நினைவகத்தில் ஏற்றிக் கொண்டு குறிப்பிட்ட கட்டளையைக்குறிய (ls) வெளியீட்டினைத் தரும்.
#ls இது listing கட்டளை
#strace ls இது listing கட்டளை கொடுத்தனால் என்னவெல்லாம் நடக்கிறது என்பதை பயனர் அறிய உதவும் கட்டளை.
ix) வன்பொருள் மேலாண்மை (hardware management)

பொதுவாக ஓர் இயங்குதளத்தில் வன்வட்டு, நினைவக அட்டைகள், குச்சி நினைவகங்கள், குறுவட்டுக்கள், அடர்குறுவட்டுக்கள், தொலை வன்வட்டுக்கள் போன்றவை கையாளப்படுகின்றன. இது அல்லாமல் காப்புப்படி எடுக்கும் பொழுது, வேறுபட்ட காப்புப்படி ஒலி

நாடாக்களோ(முந்நாட்களில்) அச்சடிக்கும் பொழுது அச்சுப்பொறிகளோ கையாளப்படுகின்றன. இதில் எந்த வகையான கருவிகளைக் கையாண்டாலும் பயனருக்குத் தோழமை கொடுக்கும் வண்ணம் வன்பொருள் மேலாண்மை செய்யப்படல் வேண்டும்.

x) வலைய மேலாண்மை **(network management)**

இது இரண்டு வகைப்படும்.

1. கம்பியுள்ள வலையம்

2. கம்பியில்லா வலையம்

இந்த இரண்டுமே வரையறுக்கப்பட்ட வலைய அடுக்குகள் மற்றும் வலையக்கருவி இயக்கிகள் கொண்டு செயல்படுகின்றன.

இந்த ஐந்து வகையான மேலாண்மைகளும் ஒன்றுக்கொன்று நெருங்கிய தொடர்புடையவை. இவை அனைத்தையும் சரிவர செயல்படுத்துவதே ஒரு திறனுள்ள இயங்குதள த்தின் வேலையாகும்.

ஒவ்வொரு மொழி பொறுத்தும் தரவு அமைவுகள் வேறுபடும். இங்கு ஷெல் நிரல்களைப் பொறுத்த அளவில் எண்கள் மற்றும் உரைகள் என இரு வகை தரவுகளே உள்ளன. உயர் நிலை மொழிகளான சி, சி++, ஜாவா மொழிகளில் நாம் ஒரு மாறியைப் பயன்படுத்த வேண்டுமெனில் அதனை நாம் வெளியிட வெண்டும். ஆனால் ஷெல் நிரலானது உயர் மொழி இல்லாததால் நாம் அவ்வாறு வெளியிடத் தேவையில்லை.

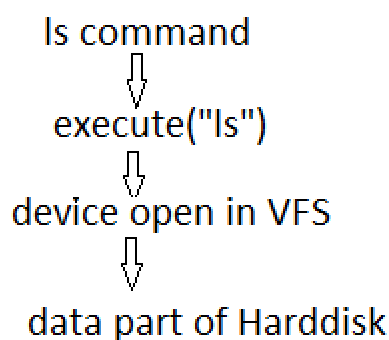
எண்கள்

இங்கு எண்களானது, இயல் எண்கள், முழுக்கள், என வேறுபடுத்தி அறியப்படுவதில்லை. அவை அந்தந்த இடங்களைப் பொறுத்து மாறுபட்டு நிற்கின்றன. அவற்றை நமது தேவைகளுக்கேற்ப பயன்படுத்திக் கொள்ளலாம்.

உரைகள்

எந்த வகையான உரையாக இருப்பினும் அவை இரட்டை மேற்கோள்குறிக்குள் எழுதப்படல், அனுகப்படல் வேண்டும். உரைகளுக்குள்ளே ஒப்பிடும் பொழுது அதே போன்று செயதல் வேண்டும்.

எண்கள், உரைகள் ஆகியவற்றைப் பற்றி விரிவாக பிறகு வரும் நிரல்களின் விளக்கத்தின் பொழுது காணலாம்.
ஐயம்: ஷெல் ஸ்கிரிப்ட்டுகள் அர்ரே (Array) எனப்படும அடுக்குமாறிகளை ஆதரிக்குமா?
தீர்வு: ஷெல்ஸ்கிரிப்ட் அர்ரே எனப்படும் அடுக்குமாறிகளை நேரடியாக ஆதரிக்காது. இருப்பினும் ஒரு பரிமாண அடுக்குமாறிகளை (one dimensional array) நாம் நேரடியாக நிகர்படுத்தி (assign) பயன்படுத்தலாம். அவை குறித்த நிரல்களை பின்னால் பார்க்கலாம். அடுக்குமாறிகள் அனைத்தும் சுழி அடிப்படை தொடக்கத்தைக் (zero-based index) கொண்டவையாகும். அடுக்குமாறிகளுக்கு வரம்பெல்லை எதுவும் கிடையாது. ஆனால் உயர்நிலை மொழிகளான சி,சி++,ஜாவா போன்று இரு பரிமாண (two dimensional), முப்பரிமாண (three dimensional), பல்பரிமாண(multi-dimensional) அடுக்குமாறிகளை ஷெல் ஸ்கிரிப்ட் ஆதரிக்காது.
ls கட்டளையின் பொழுது பின்வரும் படத்தில் உள்ளது போல் இயங்குதளத்தில் செயற்பாடுகள் நிகழும்.



இங்கு VFS என்பது Virtual File System என்பதைக் குறிக்கும். இனி எண்கள் உரைகள் ஆகியவற்றை எவ்வாறு நிரலில் பயன்படுத்தலாம் என்பதைக் காணலாம். எண்களை நிரலில் அறிக்கும் முறை:
i=5
j=6
உரைகளை நிரலில் அறிவிக்கும் முறை:
one=”This is one”
two=”This is two”
three=”3” இங்கே 3 என்ற எண் உரையாகக் கருதப்படுகிறது. பின்வரும் நிரலில் இவற்றை நாம் அழைக்கலாம்.

நிரல் **2:**

```
#!/bin/bash
echo “i value is $i”
echo “j value is $j”
echo “first string is $one”
echo “second string is $two”
echo “third string is $three”
```

ஒரு குறுநிரல்**(shellscript)** எப்படியிருக்க வேண்டும்**?**

குறுநிரலானது பின்வரும் ஐந்து முதன்மையான கொள்கைகளைக் கொண்டதாக இருக்க வேண்டும்**.**

1. குறுநிரலானது பிழைகளின்றி இருத்தல் வேண்டும் .
2. நிரலானது ஏரணப் பிழையறத் தெளிவுற அமைதல் வேண்டும்.
3. நிரலானது ஒரு குறிப்பிட்ட செயலினைச் செய்தல் வேண்டும் .
4. குறுநிரலானது தேவையற்ற செயல்களைச் செய்தல் கூடாது .
5. குறுநிரல்கள் திரும்பப் பயன்படுவதற்கு ஏற்றதாக அமைதல் வேண்டும்.

நிரல் **3:**

தற்பொழுது பொறியில் உள்நுழைந்திருக்கும் **(system login users)** பயனர்களின் பெயர்களை மட்டும் வரிசைப்படுத்தும் குறுநிரலை கீழ்க்கண்டவாறு எழுதலாம்**.**

பொதுவாக **w** அல்லது **who** ஆகிய இரண்டு கட்டளைகள் மூலம் ஒரு பொறியினுள் உள்நுழைந்திருக்கும் பயனர்களின் அனைத்து விபரங்களை அறிய முடியும் **.** கீழ்க்காணும் கட்டளை அதற்கு உதவுகிறது**.**

#who | sort | cut -d‘ ’ -f1

இங்கு **who** என்பது முதன்மைக் கட்டளை ஆகும்**. sort** என்பது வெளியீட்டினை அகரவரிசையில் அமைக்க உதவும் கட்டளையாகும்**. cut** என்பது ஒரு குறிப்பிட்ட **field** எனப்படும் நிரையினை **(column)** மட்டும் பிரிக்க உதவும் கட்டளையாகும்**. d** என்பது **delimiter or separator** என்றழைக்கப்படும் பிரிப்பான் ஆகும்**. f1** எனது முதல் நிரையினைக் **(column)** குறிக்கிறது **.** இதற்கான முழுநிரலைக் கீழ்க்காணுமாறு எழுதலாம்**.**

```
#!/bin/bash
# whos – a program which displays the login usernames of a particular system.
```

who | sort | cut -d‘ ’ -f1

[illegible]. **(vim whos.sh; chmod +x whos.sh; ./whos.sh)**

```
root@localhost:~/Desktop/scripts
File  Edit  View  Search  Terminal  Help
#!/bin/bash
# whos – a program which displays the login usernames of a particular system.

who | sort | cut -d' ' -f1

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"whos.sh" 5L, 121C written
```

```
root@localhost:~/Desktop/scripts
File Edit View Search Terminal Help
[root@localhost scripts]# cat whos.sh
#!/bin/bash
# whos – a program which displays the login usernames of a particular system.

who | sort | cut -d' ' -f1

[root@localhost scripts]# chmod +x whos.sh
[root@localhost scripts]# ./whos.sh
root
root
[root@localhost scripts]# 
```

[illegible]:

[illegible] - Basic arithmetic operations

[illegible] - Primary memory

[illegible] - Secondary memory

[illegible] - Virtual memory or swap memory

[illegible] - Temporary file system

[illegible] - Numbers, Integers, Floats, Double etc.

[illegible] - Texts, Strings

[illegible] - Printers

[illegible] - Backup tapes

[illegible] - Wired network

[illegible] - Wireless network

[illegible] - predefined network layers

வலையக்கருவி இயக்கிகள் **- network device drivers**

ஏரணம் **- logic**

தரவுகள் **- data**

வெளியீடுதல் **- declaration**

சுழி **- zero**

(கற்போம்...**)**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 3 செயல் வகைகள், நிலைகள்

பெற்றோர் குழந்தை இல்லார் உதவியர்
என்றே தளத்தில் உண்டாம் செயல்கள்
ஓட்டம் நிற்றல் உள்ளுறை உறக்கம்
குறுக்கிடு மற்றும் குறுக்கிடா நிலைகளே.
- நிரற்பா 3

நிரற்பா விளக்கம் :

நிரற்பாவினை மனப்பாடம் செய்து வைத்துக் கொண்டால் இலினக்சு இயங்குதளத்தின் செயல்களையும், அவற்றின் நிலைகளையும் எளிதில் நினைவுக்கு கொண்டுவர இயலும். இங்கு பெற்றோர் என்பது parent process ஐயும், குழந்தை child process, இல்லார் என்பது orphan process உதவியர் என்பது daemon process ஆகியவறையும் குறிப்பிடுகிறது. எனவே நான்கு முதன்மையான செயல்கள் உள்ளன. அவற்றின் நிலைகளாக ஓட்டம் (running or active state), நிற்றல் (stopping state), உள்ளுறை (dead or zombie state), உறக்கம்(sleeping state) உறக்கத்தின் துணை நிலைகளாக குறுக்கிடு உறக்கம்(interruptable sleep state), குறுக்கிடா உறக்கம்(uninterruptable sleeping state) ஆகியவை உள்ளன.
பெற்றோர் செயல் (Parent Process) என்பது ஒன்று அல்லது அதற்கு மேற்பட்ட செயல்களைத் தன் செயல் நிறைவேறுவதற்காகத் தன்னகத்தே கொண்டதாகும்.
குழந்தை செயல் (Child Process) என்பது வேறொரு செயலானது உருவாக்கிய செயலாகும்.
பெற்றோர் செயலானது முடிந்த பிறகும், குறிப்பிட்ட குழந்தை செயல் தன்னாலாகச் செயல்பட்டுக் கொண்டிருத்தல் இல்லார் செயல் (Orphan Process) அல்லது பெற்றோர் இல்லாச் செயல் எனப்படும்.
ஏதேனும் ஒரு செயலின் உதவிக்காக பின்புலத்தில் செயல்படுவது உதவியர் (Daemon Process) செயல் எனப்படும்.

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -31- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

உள்ளுறை நிலை (Zombie state) என்பது, ஒரு குறிப்பிட்ட பெற்றோர் செயலானது முடிந்த பிறகும், அதனுடைய அட்டவணையில் இருக்கும் ஒரு செயல். இது செயலின் ஒரு செத்த நிலை.
ஓட்டம் என்பது தற்போது செயலிலும், உறக்கம் என்பது முன்னர் செயலிலும் இருந்த நிலைகளாகும். நிற்றல் என்பது முடிந்த (Stop state) நிலையாகும்.

விளக்கப்படம்:
பின்வரும் விளக்கப்படங்கள் இந்த செயல்களையும், அவற்றின் நிலை மற்றும் துணை நிலைகளை விளக்கும் வண்ணம் அமைந்துள்ளன.

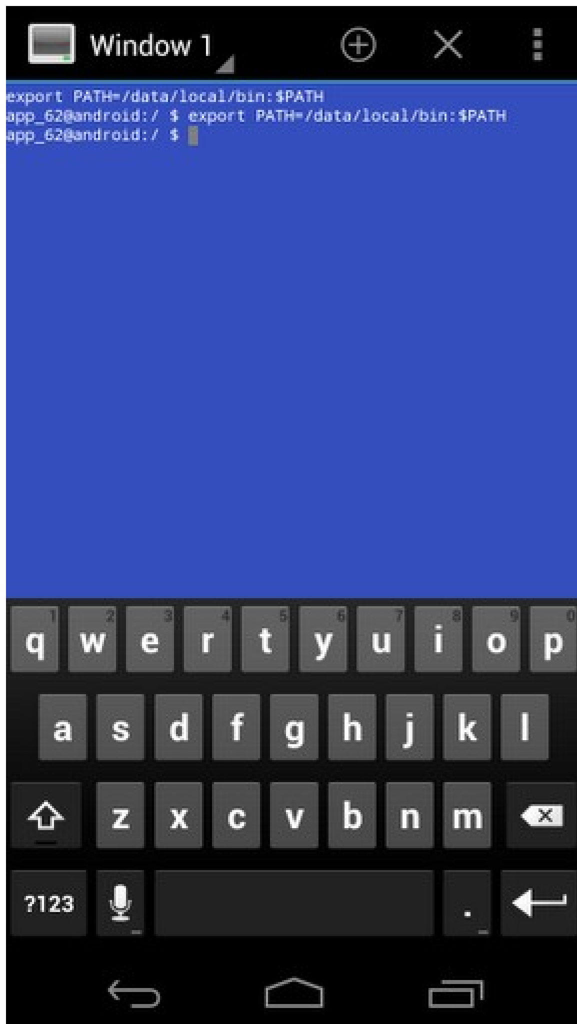
செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -32- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

Type of processes in Linux	State of processes
Parent (பெற்றோர்)	Running (ஓட்டம்)
Child (குழந்தை)	Sleeping (உறக்கம்)
Orphan (இல்லார்)	— Uninterrupted sleep (குறுக்கிடு உறக்கம்)
Daemon (உதவியர்)	— Interrupted sleep (குறுக்கிடா உறக்கம்)
	Stopping (நிற்றல்)
	Zombie (உள்ளுறை) உள்ளே உறைந்து இறந்த

uninterruptible sleep
Signal
Stopped
uninterruptible sleep
Signal
wake
Running
Exit
interruptible sleep
Zombie
interruptible sleep
Signal
Process Created

[illegible]: [illegible]?

[illegible]: **Terminal emulator** [illegible] **echo** [illegible]. http://www.appsapk.com/android-terminal-emulator/ [illegible]. [illegible] **root** [illegible] **hack** [illegible].



Complete Linux Installer http://www.appsapk.com/complete-linux-installer/ என்ற தொடுப்பில் உள்ள பொதியினை பதிவிறக்கி நிறுவிக்கொண்டால், அனைத்து வகையான குறுநிரல்களையும் எழுதி மகிழலாம். ஆனால் இது புதியவர்களுக்கு உகந்ததல்ல. ஏதேனும் தவறான நிரல் எழுதி கணிப்பொறியில் பழுதேற்பட்டால், அதை மீண்டும் நிறுவிக் கொள்ளலாம். ஆனால் கைபேசியில் பழுதேற்பட்டால் மீண்டும் அனைத்தும் மீண்டு வருமா என்பது மிகப்பெரும் ஐயமே.

சின்ன சின்ன நிரல்களை எழுதி மகிழலாம். பெரிய நிரல்களுக்கு இது போன்ற ஆண்ட்ராய்டு செயலிகள் வழிவகுத்தாலும், அவை தொழிற்நுட்ப வகையில் பரிந்துரைக்கப்படுவதில்லை. **(Professionally not recommended)**

குறுநிரல் **3: (Terminal Emulator** மூலம் இயக்கலாம்**)**

```
#!/bin/bash
#Script that displays today’s date
echo "Today's date is:"
 date +"%A, %B %-d, %Y"
```

நிரல் விளக்கம்:

Terminal emulator மூலம் இயக்கும் பொழுது, **cd sdcard** என்று கொடுத்து, ஒரு அடைவினை **(directory/folder)** உருவாக்கிக் கொள்ளவும். **mkdir script** என்ற கட்டளை இதற்கு உதவும். பின்பு **cd script** என்பதற்குள் சென்று வேண்டிய நிரல்களை எழுத்தத் தொடங்கலாம். கைபேசியை **root** செய்தால்தான் உரைத் தொகுப்பான்களை கட்டளை வரியில் பயன்படுத்த முடியும். இல்லையில் **quick office** மூலம் நிரலைத் தொட்டெழுதி/தட்டெழுதி அதை **Terminal emulator** இல் இயக்கலாம். இதன் வெளியீடானது, நிரல் இயக்கிய கிழமை, மாதம் தேதி, ஆண்டு ஆகியவற்றை வெளியிடும். எடுத்துக்காட்டாக , நிரல் அக்டோபர் **1** அன்று இயக்கப்பட்டால் வெளியீடு இவ்வாறு கிடைக்கும். **Today’s date is: Wednesday, October 1, 2014**

குறுநிரல் **4 : (Terminal Emulator** மூலம் இயக்கலாம்**)**

இந்த நிரல், எவ்வாறு ஷெல் ஸ்கிரிப்ட் மூலமாக ஒரு எச்டிஎம்எல் கோப்பினை உருவாக்கப் பயன்படுகிறது என்று பார்க்கலாம். எடுத்துக்காட்டாக

எச்டிஎம்எல் கோப்பு பின்வருமாறு வைத்துக் கொள்வோம். இதை எவ்வாறு ஷெல் ஸ்கிரிப்ட் மூலமாகச் செய்யலாம் எனப் பார்க்கலாம்.

```
<HTML>
<HEAD>
   <TITLE>
   The title of your page
   </TITLE>
</HEAD>

<BODY>
   Your page content goes here.
</BODY>
</HTML>
```

குறுநிரல் **4 -** பதிப்பு **1: (Terminal Emulator** மூலம் இயக்கலாம்**)**

```
#!/bin/bash
# make_page - A script to produce an HTML file
echo "<HTML>"
echo "<HEAD>"
echo "  <TITLE>"
echo "  The title of your page"
echo "  </TITLE>"
echo "</HEAD>"
echo ""
echo "<BODY>"
echo "  Your page content goes here."
echo "</BODY>"
echo "</HTML>"
```

இந்த நிரலை இயக்கும் பொழுது, **(**கோப்பினை இப்பெயரில் சேமிக்க வேண்டும் என்பதறிக**.) main_page.sh > filename.html** என்று கொடுத்தால் அதே அடைவில் **HTML** கோப்பு உருவாகும். நாம் பார்த்த வெறும் **echo** கட்டளை கொண்டு அருமையாக ஒரு **HTML** நிரல் தயாரிக்கப்பட்டுவிட்டது **.**

நிரல் **4 -** பதிப்பு **2 (Terminal Emulator** மூலம் இயக்கலாம்**)**

இதே நிரலை **echo** கட்டளை இல்லாமலேயே கீழ்க்காணுமாறு எழுதி இயக்கலாம். இதில் **cat** கட்டளை ஒரு கோப்பினை உருவாக்கப் பயன்படுகிறது. **_EOF_** என்பது **End Of File** என்பதைக் குறிக்கிறது.

```
#!/bin/bash
# make_page - A script to produce an HTML file
cat << _EOF_
<HTML>
<HEAD>
    <TITLE>
    The title of your page
    </TITLE>
</HEAD>

<BODY>
    Your page content goes here.
</BODY>
</HTML>
_EOF_
```

நிரலை இயக்கும் நேரம், **main_page.sh > filename.html** என்று கொடுத்தால் ஒரு **HTML** கோப்பு உருவாகும்.

கலைச்சொற்கள்: (Technical terms)

பெற்றோர் – **Parent process**

குழந்தை – **child process**

இல்லார் – **orphan process**

உதவியர் – **daemon process**

ஓட்டம் – **running state**

நிற்றல் – **stopping state**

உள்ளுறை – **zombie state**

உறக்கம் – **sleeping state**

குறுக்கிடு – **interruptable sleep state**

குறுக்கிடா – **uninterruptable sleep state**

அடைவு – **Folder or directory**

(கற்போம்...)

செம்மொழியில்கற்போம்ஷெல்ஸ்கிரிப்ட் – 4 இயங்குதளத்தின்ஏற்றநிலை, மாறிகளைப் பயன்படுத்துதல்

இயங்கு தளத்தின் ஏற்ற நிலையில் முதன்மைக் கோப்பாம் மையம் தொடக்கம்
ஏழ்நிலை தெரிந்து எடுத்தி யாங்கு
இணைப்புக் கோப்பு எழிலுற வருமே.
-நிரற்பா 4

நிரற்பா விளக்கம் :

லினக்சு இயங்குதளத்தின் ஏற்ற நிலையில் அதாவது boot ஆகும் பொழுது முதலிலே மையக்கோப்பான kernel கோப்பு ஏற்றப்படுகிறது. பின்பு தொடக்கம் எனப்படும் initrd.img கோப்பு ஏற்றப்படுகிறது. பின்பு இயங்குதளத்தின் ஏழு நிலைகளில் எந்த நிலையானது அமைக்கப்பட வேண்டும் என்று /etc/inittab கோப்பில் கட்டளை இருக்கிறதோ அந்த நிலையில் இயங்குதளம் தொடங்கப் படுகிறது. வன்வட்டு அமைவுகளைக் கொண்ட /etc/fstab என்ற இணைப்புக் கோப்பில் உள்ளவாறு இயங்குதளம் தனது வேலைகளைச் செய்ய தொடங்குகிறது.

மாறிகளைப்பயன்படுத்துதல்: (Variable usage)
மாறிகளில் பொதுவாக இரண்டு வகைகள் உண்டு. ஒன்று env என்ற கட்டளை மூலம் கிடைக்கும் environment variable எனப்படும் சூழல் மாறிகள். இரண்டாவது பயனர் தனது நிரலில் வரையறை செய்யும் மாறிகள் (user-defined variables). # வரியில் env என்று கொடுக்க அனைத்து சூழல் மாறிகளும் கிடைக்கப் பெறும்.

ஐயம்:ஷெல்ஸ்கிரிப்ட்மூலம்தீங்குவிளைவிக்கும்நிரல்களைஎழுதமுடியுமா?

தீர்வு:கருத்துஅடிப்படையில்பார்த்தால்தீங்குவிளைவிக்கும்நிரல்களை (destruction scripts)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -39- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

நேரடியாகவோமறைமுகமாகவோஎழுதஇயலும் .ஒருபொறியின்மூலப்பயனராக **(root user)**இருப்பின்பொறிக்குஎதுதீங்குசெய்யும்என்றுஅறிந்துநிரலமைப்பதுஇன்றி யமையாதது மூலப்பயனர்ஏதேனும்தவறுசெய்தால்அதைவேறுயாராலும்மீளமைக்க**(undo)** முடியாது.

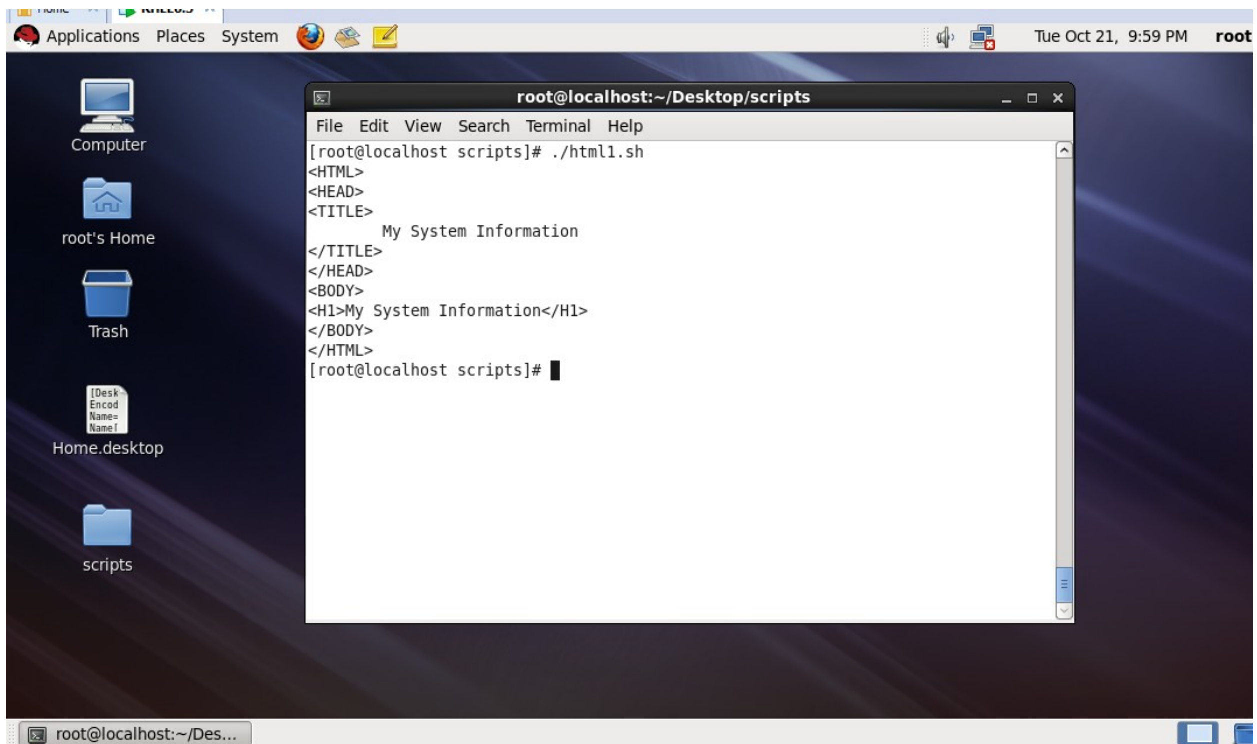
மாற்றீடுதல்

நிரல் **5: (Terminal Emulator** ல்எழுதிஇயக்கலாம்**)**

இந்த நிரலில் **title** என்பது மாறியாக அமைக்கப்பட்டுள்ளது. அந்த மாறியினை அழைக்கும் பொழுது **$title** என்று அமைக்கப்படவேண்டும்.

```
#!/bin/bash
# make_page - A script to produce an HTML file
title="My System Information"
cat<<- _EOF_
<HTML>
<HEAD>
<TITLE>
	$title
</TITLE>
</HEAD>
<BODY>
<H1>$title</H1>
</BODY>
</HTML>
_EOF_
```

நிரலின் வெளியீடானது பின்வருமாறு அமையும். (chmod +x niral5.sh; ./niral5.sh)



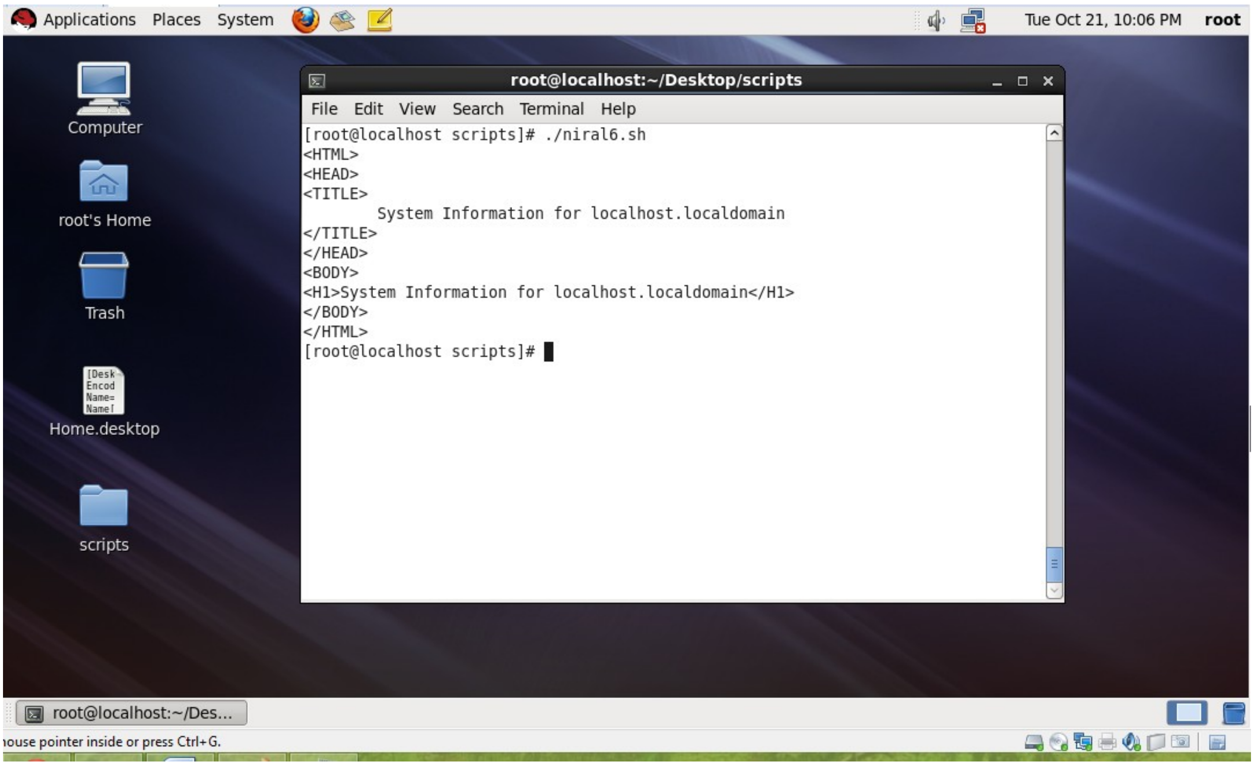
சூழல்மாறிகள்:

பின்வரும் நிரல்களில் சூழல் மாறிகளை எங்ஙனம் பயன்படுத்தலாம் என்பதைக் காண்போம்.
நிரல் 6: (Root செய்யப்பட்ட Terminal Emulator ல்எழுதிஇயக்கலாம்)
இங்கு $HOSTNAME என்பது பொறியின் பெயராகிய environment variable ஐக் குறிக்கிறது.
#!/bin/bash
# make_page - A script to produce an HTML file
title="System Information for"

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -41- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

```
cat<<- _EOF_
<HTML>
<HEAD>
<TITLE>
        $title $HOSTNAME
</TITLE>
</HEAD>
<BODY>
<H1>$title $HOSTNAME</H1>
</BODY>
</HTML>
_EOF_
```

நிரலின் வெளியீடானது பின்வருமாறு அமையும். **(chmod +x niral6.sh; ./niral6.sh)**



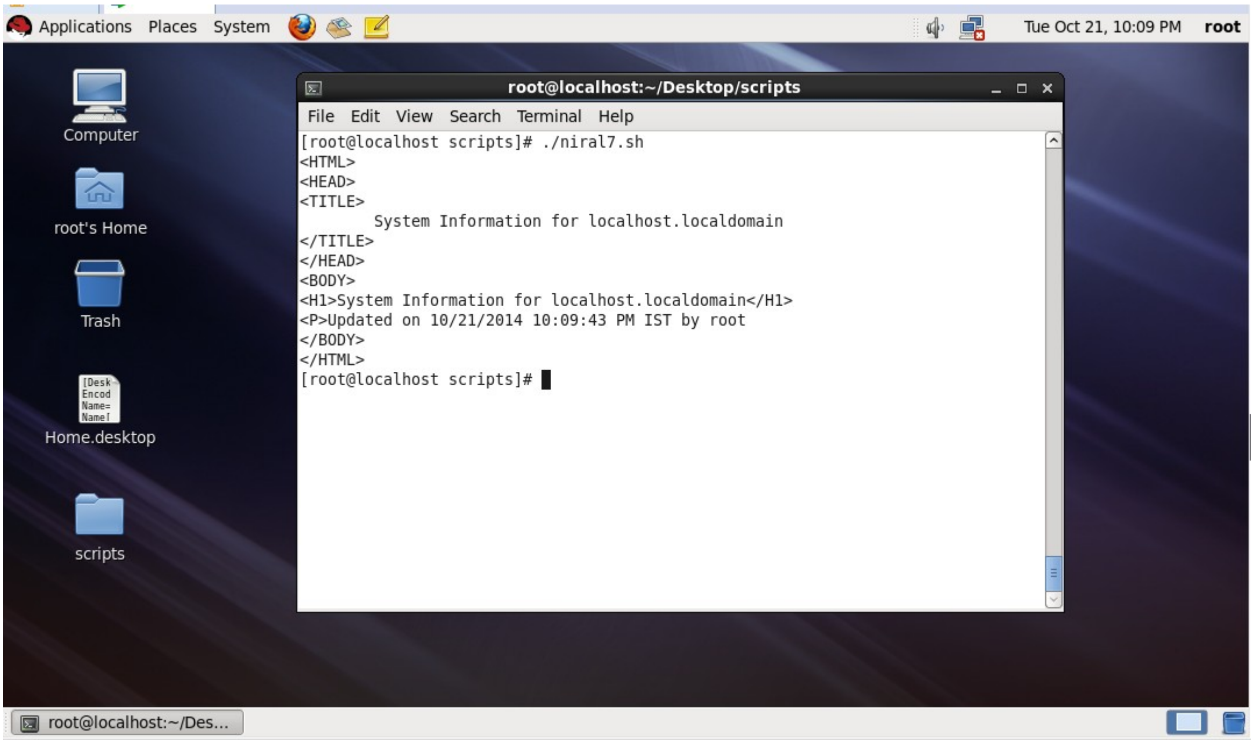
நிரல் **7: (Root** செய்யப்பட்ட **Terminal Emulator** ல்எழுதிஇயக்கலாம்**)**

இந்நிரலில் **date** என்ற கட்டளையும், **HOSTNAME, USER** ஆகிய சூழல் மாறிகளும் பயன்படுத்தப் பட்டிருக்கின்றன .

```
#!/bin/bash
# make_page - A script to produce an HTML file
TITLE="System Information for $HOSTNAME"
RIGHT_NOW=$(date +"%x %r %Z")
TIME_STAMP="Updated on $RIGHT_NOW by $USER"
cat<<- _EOF_
<HTML>
<HEAD>
<TITLE>
	$TITLE
</TITLE>
</HEAD>
<BODY>
<H1>$TITLE</H1>
<P>$TIME_STAMP
</BODY>
</HTML>
_EOF_
```

கீழ்க்கண்ட கட்டளை வரிகளைக் கொண்டு இயக்க , நிரலின் வெளியீடு படத்தில் உள்ளது போல் கிடைக்கும்.

```
#chmod +x niral7.sh
#./niral7.sh
```



இயங்கு தளத்தில் ஏழு வகை நிலைகள் அவற்றின் பயன்பாடுகளை அடுத்த தலைப்பில் பார்க்கலாம்.

கலைச்சொற்கள்: (Technical terms)

இயங்குதளம் – **Operating system**

ஏற்றநிலை – **Booting stage**

முதன்மைக்கோப்பு – **First file**

மையம் – **kernel (core of the operating system)**

தொடக்கம் – **initrd file**

ஏழ்நிலை – **seven run levels of the operating system**

இணைப்புக்கோப்பு **- /etc/fstab file (file system table configuration file)**

மூலப்பயனர், வேர்ப்பயனர் **- root user**

மீளமைத்தல் **- undo**

அழிப்புநிரல் **- destruction script**

மாறிகள் **- variables**

சூழல்மாறிகள் **- environment variables**

மாற்றீடுதல் **- Substitution**

(கற்போம்…**)**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 5 இயங்குதளத்தின் ஏழு ஓடு நிலைகள்

நிறுத்தம் ஒற்றைப் பயனர் வலையிலா
பல்பய னர்வலை யுள்பல் பயனர்
பிற்கால உயர்வு பயனர் வரைகலை
மறுதொடக் கம்இயங் குநிலை ஏழே.
- நிரற்பா 5

நிரற்பா விளக்கம் :

இப்பாவில் இயங்குதளத்தில் உள்ள ஏழு நிலைகள் விளக்கப்பட்டுள்ளன. நிறுத்தம் என்பது init 0 என்ற நிலையைக் குறிக்கிறது. இது shutdown செய்யப்பட்ட நிலையாகும். ஒற்றைப்பயனர் என்பது single user என்பதையும் வலையிலா பல்பயனர் என்பது multiuser without network என்பதையும், வலையுள் பல்பயனர் என்பது multiuser with network என்பதையும், பிற்கால உயர்வு future enhancement or unused என்பதையும், பயனர் வரைகலை என்பது GUI-Graphical User Interface என்பதையும், மறுதொடக்கம் என்பது restart என்பதையும் குறிக்கிறது.
init 0 – shutdown
init 1 – single user mode
init 2 – multi user mode without network
init 3 – multi user mode with network
init 4 – unused for future enhancement purpose
init 5 – XII or GUI mode
init 6 – reboot

கீழே உள்ள படம் இலினக்சின் ஏழு ஓடு நிலைகளையும் விளக்கும் வண்ணம் உள்ளது.

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -46- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

#vim /etc/inittab □□□□ □□□□□□□□ □□□□□□□ □□□□ □□□□□□□□. □□□□□□□□□□□ Red Hat Enterprise Linux □□□ □□□ □□□□ 3 □□□□□□ 5 □□□□ □□□□□□□□□□. Ubunt □□□ □□□□□□□ 3 □□ □□□□□□□□□□.

```
#
# inittab       This file describes how the INIT process shou
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.m
#               Modified for RHS Linux by Marc Ewing and Donn
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:


# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
```

□□□□□ 8

```
#!/bin/bash
# make_page - A script to produce an HTML file
TITLE="System Information for $HOSTNAME"
RIGHT_NOW=$(date +"%x %r %Z")
TIME_STAMP="Updated on $RIGHT_NOW by $USER"
cat <<- _EOF_
    <HTML>
    <HEAD>
        <TITLE>
        $TITLE
        </TITLE>
    </HEAD>
    <BODY>
    <H1>$TITLE</H1>
```

```
    <P>$TIME_STAMP
    </BODY>
</HTML>
_EOF_
```

[illegible] RIGHT_NOW, TIME_STAMP [illegible]. [illegible] (updated) [illegible].

[illegible] 9:

```
#!/bin/bash
# system_page - A script to produce an system information HTML file
##### Constants
TITLE="System Information for $HOSTNAME"
RIGHT_NOW=$(date +"%x %r %Z")
TIME_STAMP="Updated on $RIGHT_NOW by $USER"
##### Functions
function system_info
{
    # Temporary function stub
    echo "function system_info"
}
function show_uptime
{
    # Temporary function stub
    echo "function show_uptime"
}
function drive_space
{
    # Temporary function stub
    echo "function drive_space"
}
function home_space
```

```
{
    # Temporary function stub
    echo "function home_space"
}
##### Main
cat <<- _EOF_
  <html>
  <head>
      <title>$TITLE</title>
  </head>
  <body>
      <h1>$TITLE</h1>
      <p>$TIME_STAMP</p>
      $(system_info)
      $(show_uptime)
      $(drive_space)
      $(home_space)
  </body>
  </html>
_EOF_
function show_uptime
{
    echo "<h2>System uptime</h2>"
    echo "<pre>"
    uptime
    echo "</pre>"
}
function drive_space
{
    echo "<h2>Filesystem space</h2>"
    echo "<pre>"
    df
```

```
   echo "</pre>"
}
function home_space
{
   echo "<h2>Home directory space by user</h2>"
   echo "<pre>"
   echo "Bytes Directory"
   du -s /home/* | sort -nr
   echo "</pre>"
}
function system_info
{
   echo "<h2>System release info</h2>"
   echo "<p>Function not yet implemented</p>"
}
```

நிரல் துண்டுகள் (Functions):

இந்நிரலில் ஃபங்ஷன்கள் என்னும் நிரல் துண்டுகள் அறிமுகப்படுத்தப்பட்டுள்ளன. அதாவது ஒரு பெரிய நிரலை சிறிதுசிறிதாக உடைத்து, ஒரு சிறிய வேலையினைச் செய்து முடிப்பதே நிரல் துண்டு எனப்படுகிறது. **(a function is a piece of a program, which is used to perform a particular task in the main program)**

```
$(system_info)
$(show_uptime)
$(drive_space)
$(home_space)
```

இங்கு நான்கு நிரல் துண்டுகள் பயன்படுத்தப்பட்டுள்ளன . **system_info** பொறியினைப் பற்றிய செய்திகளைத் தருவதாக அமைந்திருக்கிறது. **show_uptime** என்னும் நிரல் துண்டானது, பொறியானது எவ்வளவு நேரம் செயல்பாட்டிலிருக்கிறது என்ற செய்தியைத் தருகிறது. **drive_space** நிரல் துண்டில் வன்வட்டின் அளவு காலியிடம் போன்றவை வரையறுக்கப்பட்டுள்ளது.

home_space என்ற துண்டில் user’s home directory ன் அளவு விளக்கப்பட்டுள்ளது.

நிரல் துண்டின் பயன்பாடுகள் (usage of functions in shell script):

1. ஒரு குறிப்பிட்ட வேலையினை மட்டுமே செய்யப்பயன்படுகிறது.
2. மீண்டும் மீண்டும் எழுத வேண்டிய நிரல்வரிகளை ஒரே முறை எழுதி மீண்டும் மீண்டும் முதன்மை நிரலில் இருந்து அழைக்கலாம்.
3. நிரலர்களின் பணி எளிதாகிறது.
4. பொறியும் எளிதில் நிரல்வரிகளைப் புரிந்து குழப்பமின்றி செயல்பட உதவுகிறது.
5. வருங்கால நிரல் மேம்பாட்டிற்கு உதவுகிறது.
6. நிரலினை புதியவர்கள் படிக்கும் பொழுது, எளிதில் புரியும் வண்ணம் அமைகிறது.

கலைச்சொற்கள் :

நிறுத்தம் – init 0 (shutdown or halt)
ஒற்றைப் பயனர் – single user (run-level 1)
வலையிலா பல்பயனர் – multiuser without network (run-level 2)
வலையுள் பல்பயனர் – multiuser with network (run-level 3)
பிற்கால உயர்வு – future enhancement (run-level 4)
பயனர் வரைகலை – Graphical user interface (GUI run-level 5) மறுதொடக்கம் – restart (run-level 6)
நிரல் துண்டு - functions
(கற்போம்)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -51- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 6
செயற்பாடு அல்லது நிரல்துண்டு (Functions)

பெருஞ்செயல் ஒன்றைப் பிரித்து, படுத்தும்
சிக்கல் வரிகளைச் சிறுதுண் டாக்கி
வேண்டும் பொழுதில் விரைவாய் அழைக்கக்
கட்டளை வரிகுறைப் பதாம்நிரல் துண்டே.
- நிரற்பா 6

நிரற்பா விளக்கம்:

பெரிய வேலையைச் செய்யக் கூடிய நிரலினை சிறிய துண்டுகளாக்கி தேவையான பொழுது அழைத்துப் பயன்படுத்திக் கொள்வதால் கட்டளை வரிகள் குறையும் . விரைவாகச் செயல்கள் முடியும்.

நிரல் **10:**

```
#!/bin/sh
# A simple script with a function...
add_a_user()
{
 USER=$1
 PASSWORD=$2
 shift; shift;
 # Having shifted twice, the rest is now comments ...
 COMMENTS=$@
 echo "Adding user $USER ..."
 echo useradd -c "$COMMENTS" $USER
 echo passwd $USER $PASSWORD
 echo "Added user $USER ($COMMENTS) with pass $PASSWORD"
}
###
```

```
# Main body of script starts here
###
echo "Start of script..."
add_a_user bob letmein Bob Holness the presenter
add_a_user fred badpassword Fred Durst the singer
add_a_user bilko worsepassword Sgt. Bilko the role model
echo "End of script..."
```

நிரல் விளக்கம்:

இந்நிரலில் பயனரையும் அதற்குரிய கடவுச்சொல்லையும் எவ்வாறு நிரல் துண்டு கொண்டு அமைப்பதைக் காண்கின்றோம். ஒரு சிறிய நிரல் துண்டானது மீண்டும் மீண்டும் அழைக்கப்பட்டு பல்வேறு பயனர்களையும் அவற்றிற்குரிய கடவுச்சொற்களையும் எளிதில் அமைக்க வழிவகுக்கிறது. கீழே இந்நிரலில் அழைக்கப்பட்டுள்ள மாற்றிடும் மாறிகளைக் காண்கின்றோம் . இங்கு ஆறு மாறிகள் கையாளப்படுகின்றன. அவை பின்வருமாறு

$1=bob

$2=letmein

$3=Bob

$4=Holness

$5=the

$6=presenter

நிரல் **11:**

```
#!/bin/sh
myfunc()
{
  echo "I was called as : $@"
  x=2
}
### Main script starts here
echo "Script was called with $@"
x=1
echo "x is $x"
```

```
myfunc 1 2 3
echo "x is $x"
```

நிரல் விளக்கம்:

இங்கு **myfunc()** என்பது நிரல் துண்டாகும் . அது முதன்மை நிரலில் **(main program)** இருந்து அழைக்கப்படுகிறது. இந்நிரல் மூலம் நாம் எவ்வாறு ஓரே மாறியானது முதன்மை நிரலிலும், துண்டு நிரலிலும் மாறிமாறி அழைக்கப்படுகின்றது என்பதைக் காண்கிறோம்.

நிரல் **12:**

```
#!/bin/sh
myfunc()
{
  echo "\$1 is $1"
  echo "\$2 is $2"
  # cannot change $1 - we'd have to say:
  # 1="Goodbye Cruel"
  # which is not a valid syntax. However, we can
  # change $a:
  a="Goodbye Cruel"
}

### Main script starts here
a=Hello
b=World
myfunc $a $b
echo "a is $a"
echo "b is $b"
```

நிரல் விளக்கம்:

நிரல் **11** இல் இருப்பது போன்றே இந்நிரலிலும் நிரல் துண்டின் மூலமாக மாறியினை எப்படி வேறுவகையாகப் பயன்படுத்துவது என்று விளக்கப்பட்டுள்ளது . ஆகவே மேற்கூறிய, நிரல்களின் மூலமாக கொடுக்கப்பட்ட உள்ளீடுகள் எவ்வாறு வேறுவகையாக நிரல் வெளியீட்டில் கிடைக்கின்றன என்பது அறியப்படுகின்றது .

நிரல்களை இயக்கும் முறை: நிரல்களின் வெளியீடுகள் எதுவும் இங்கு கொடுக்கப்படவில்லை. நிரல்களை எப்பொழுதும் போல் உரைத்திருத்தியில் எழுதி filename.sh என்று சேமித்து./filename.sh என்று கொடுக்க வெளியீடு கிடைக்கும். இந்நிரல்களை Terminal emulator இல் இயக்கிப்பார்ப்பதைக் காட்டிலும், இலினக்சு இயங்குதளத்தில் இயக்கிப்பார்ப்பது நல்லது. இலினக்சு இயங்குதளத்தில் நாம் பயன்படுத்தும் பல கட்டளைகள் குறுநிரல்கள் அனைத்திலும் நிரல் துண்டுகள் மிகுதியாகப் பயன்படுத்தப்பட்டிருக்கின்றன. நிரல்களின் தவறுகளை வரிவரியாய்ப் பார்க்க sh –vx./filename.sh என்ற கட்டளைவரி கொண்டு இயக்கிப்பார்க்கலாம்.

கலைச்சொற்கள் :

செயற்பாடு – functions
நிரல்துண்டு – functions
சிக்கல்வரிகள் – complex instructions
பெருஞ்செயல் – big task or complex task
முதன்மை நிரல் – main program
உரைத்திருத்தி – text editor

(கற்போம்...)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -55- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

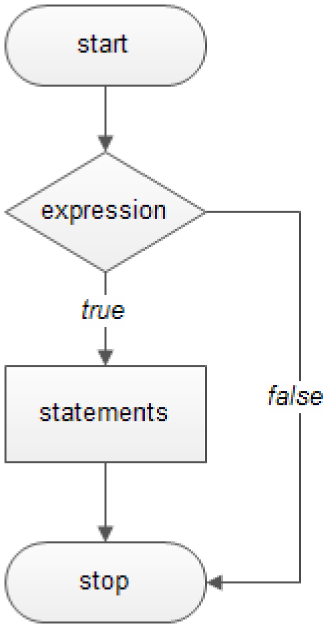
செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் 7 - கட்டுப்பாட்டு அமைவுகள் (if condition)

இருந்தால் பிறகு இஃதே நிகழும் இலையெனில் மற்றவை இயங்கும்; இருந்தால்
கூடு இல்லை; அடுக்கு இருந்தால் உண்டு; இருந்தால் ஏணியும் உண்டே. -நிரற்பா 7

நிரற்பா விளக்கம் :

இருந்தால் என்பது இப்பாடலில் if condition ஐக் குறிக்கிறது. If condition சரியாக இருந்தால் ஒன்றும், தவறாக இருந்தால் பிரிதொன்றும் நடக்கும். கூடு என்பது nested if statement ஐக்குறிக்கிறது. உயர்நிலை மொழிகளான சி,சி++ உள்ளது போன்று இதில் nested if statement கிடையாது. அடுக்கு இருந்தால் என்பது elif statement ஐக்குறிக்கிறது. அடுக்கடுக்கான elif statement கள் சேர்ந்து இருந்தால் ஏணி அதாவது elif ladder ஐ உருவாக்குகிறது. ஏற்கனவே சி, சி++ உயர்நிலை மொழி கற்றவர்களுக்கு இது எளிதில் புரியும். மற்றவர்களுக்குப் புரியும் வண்ணம் கீழே படம் கொடுக்கப் பட்டுள்ளது.

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -56- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,



குறிப்பிட்ட கட்டளையானது சரிபார்க்கப்பட்டு , சரியெனில் ஒன்று அல்லது ஒன்றுக்கு மேற்பட்ட கட்டளைவரிகள் இயக்கப்படும். தவறெனில் பிறிதொரு வகையான கட்டளை அல்லது ஒன்றுக்கு மேற்பட்ட கட்டளைவரிகள் இயக்கப்படும்.

கீழ்க்காணும் அமைப்பு இதனை எளிதாக விளக்குகிறது .

முதல் வகையான **if condition syntax:**

 # First form

if *condition* ; then

    commands

fi

இரண்டாம் வகையான **if condition syntax:**

 # Second form

if *condition* ; then

    commands

else

    commands

fi

மூன்றாம் வகை **if condition syntax:**

 # Third form

```
if condition ; then
    commands
elif condition ; then
    commands
fi
```

முதல்வகையான இருந்தால் கட்டளையினை, if statement இல்லாமலும் கையாளலாம். test என்ற keyword இங்கு if க்கு மாற்றாகப் பயன்படுத்தப்பட்டிருக்கிறது .

இரண்டாவது # Second form இல் []சதுர அடைப்புக்குறிகள் பயன்படுத்தப்பட்டிருக்கின்றன.

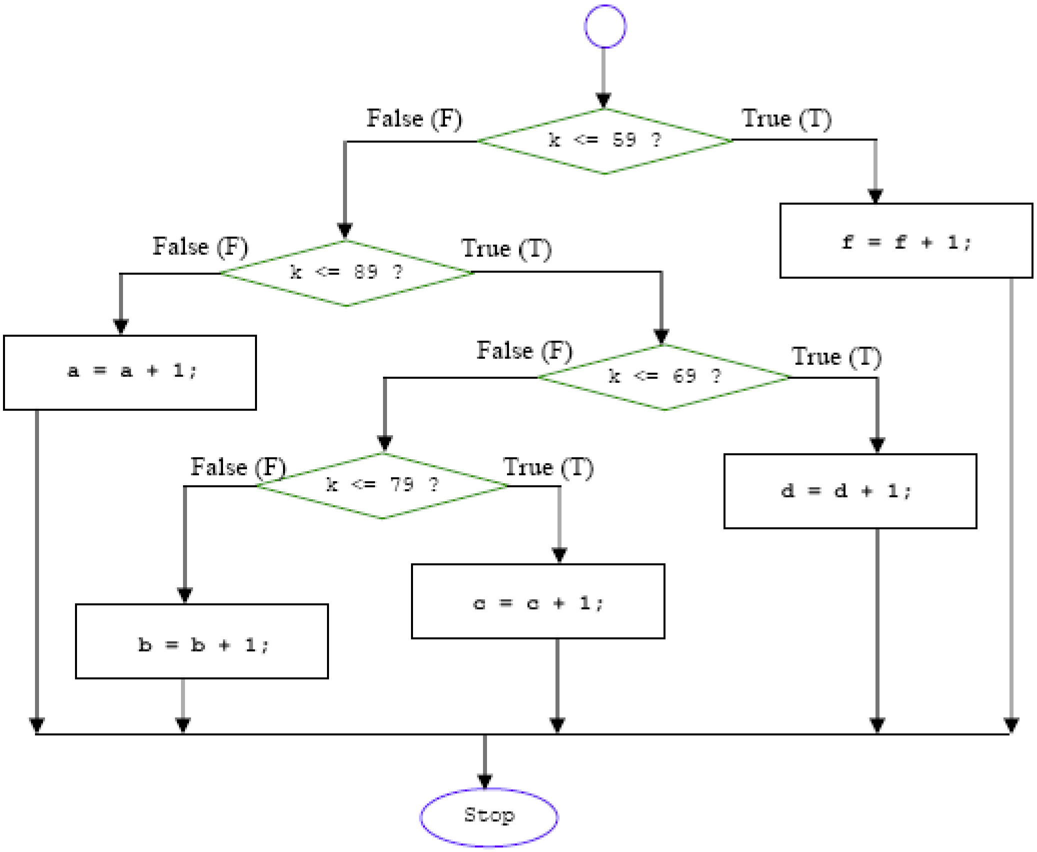
```
# First form

test expression

# Second form

[ expression ]
```

எளிய எடுத்துக்காட்டு**:**

நிரல் **13** பதிப்பு **1:**

```
if [ -f .bash_profile ]; then
    echo "You have a .bash_profile. Things are fine."
else
    echo "Yikes! You have no .bash_profile!"
fi
```

இங்கு இரண்டாம் வகையான **if condition** கையாளப்பட்டுள்ளது. **-f** என்பது குறிப்பிட்ட கோப்பானது உள்ளதா இல்லையா என்பதற்காக அமைக்கப்பட்டுள்ளது. **fi** என்பது **if condition** முடிவடையும் பொழுது கொடுக்க வேண்டிய அமைப்பாகும். உயர் நிலை மொழிகள் போன்று **{ } braces** இங்கு பயன்படுத்தப்படுவதில்லை. **then** என்கிற **keyword** ம் இன்றியமையாதது.



மேலேயுள்ள படத்தில் அடுக்கடுக்கான **if statement** கள் கையாளப்பட்டுள்ளன. இவை **decision making** எனப்படும் தீர்வறிவதறிவதற்காக பயன்படுத்தப்படுகின்றன. இது இருந்தால் ஏணி **elif ladder** என்றழைக்கப்படுகிறது.

பின்வரும் அட்டவணையானது, எவ்வாறு நாம் **if statement** இல் ஒப்பிட்டுப் பார்த்து , முடிவெடுப்பது என்பதை விளக்குகிறது.

Expression	**Description**
-d *file*	**True if *file* is a directory.**
-e *file*	**True if *file* exists.**
-f *file*	**True if *file* exists and is a regular file.**
-L *file*	**True if *file* is a symbolic link.**

-r *file*	**True if *file* is a file readable by you.**
-w *file*	**True if *file* is a file writable by you.**
-x *file*	**True if *file* is a file executable by you.**
file1* -nt *file2	**True if *file1* is newer than (according to modification time) *file2***
file1* -ot *file2	**True if *file1* is older than *file2***
-z *string*	**True if *string* is empty.**
-n *string*	**True if *string* is not empty.**
string1* = *string2	**True if *string1* equals *string2.***
string1* != *string2	**True if *string1* does not equal *string2.***

[illegible] 13 [illegible] 2:

```
if [ -f .bash_profile ]
then
    echo "You have a .bash_profile. Things are fine."
else
    echo "Yikes! You have no .bash_profile!"
fi
```

[illegible] 13 [illegible] 3:

```
if [ -f .bash_profile ]
then echo "You have a .bash_profile. Things are fine."
else echo "Yikes! You have no .bash_profile!"
fi
```

[illegible].

[illegible] 14

```
if [ $(id -u) = "0" ]; then
    echo "superuser"
```

```
fi
```

[illegible] 15

```
if [ $(id -u) != "0" ]; then
    echo "You must be the superuser to run this script" >&2
    exit 1
fi
```

[illegible], [illegible].

[illegible] 16

```
function home_space
{
    # Only the superuser can get this information

    if [ "$(id -u)" = "0" ]; then
        echo "<h2>Home directory space by user</h2>"
        echo "<pre>"
        echo "Bytes Directory"
            du -s /home/* | sort -nr
        echo "</pre>"
    fi

}   # end of home_space
```

[illegible] 17

```
#!/bin/bash

number=1

if [ $number = "1" ]; then
    echo "Number equals 1
```

fi

நிரல் **18**

```
#!/bin/bash

number=1

if [ $number = "1" ]; then
    echo "Number equals 1"
else
    echo "Number does not equal 1"
fi
```

நிரல் **19**

```
#!/bin/bash

number=1

set -x
if [ $number = "1" ]; then
    echo "Number equals 1"
else
    echo "Number does not equal 1"
fi
set +x
```

நிரல்களை வழக்கம்போல் உரைத்திருத்தியில் எழுதி இயக்கிப்பார்க்கவும் .

கலைச்சொற்கள்:

If condition statement - இருந்தால் கட்டளைவரி

Elif statement - இலையெனில் கட்டளைவரி

Nested if statement - [illegible]

Elif ladder - [illegible]

([illegible]...)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட்– 8

சுழற்சி அல்லது ஆகக்கட்டளை **(for loop)**

நிரல்களில் பயன்படும் நேரிய அறிக்கை
சுணக்கம் இன்றி சுழற்சியிற் செயற்பட
தேவைச் செயல்கள் திரும்பச் செய்யும்
ஆகக் கட்டளை கொண்டு எழுதவே.
- நிரற்பா 8

நிரற்பாவிளக்கம்:

ஒரு நிரலில் பயன்படக்கூடிய கட்டளை வரிகள் **(program statements),** தொடர் செயல்பாடுகள் தங்குதடையின்றி **(சுணக்கம் இன்றி)** விரைவாகச் செயல்பட, குறிப்பிட்ட கட்டுப்பாட்டு செயல்முறை நிறைவடைந்தால் **(get satisfied the given condition)** தொடர்ந்து சுழற்சிமுறையில் **(loop)** ஒன்று அல்லது ஒன்றிற்கு மேற்பட்ட கட்டளைகள் **(single or multiple statements)** இயங்க ஆகக் கட்டளையானது **(for statement)** உதவுகிறது.

இருந்தால் கட்டளை எடுத்துக்காட்டுகள் **(If statement examples)**

நிரல் **20:**

```
#!/bin/bash

echo -n "Hurry up and type something! > "
if read -t 3 response; then
echo "Great, you made it in time!"
else
echo "Sorry, you are too slow!"
fi
```

இங்கு **read** கட்டளையில் **-t 3** என்று கொடுக்கப்பட்டுள்ளது. இங்கு பொறியானது **(system)** மூன்று நொடிகள் பயனரின் உள்ளீட்டிற்காக **(user input)**

காத்திருக்கும் . பயனர் சரியான நேரத்தில் தட்டச்சு செய்தால் **if condition satisfied** ஆகி **Great, you made it in time!** என்ற வெளியீடு கிடைக்கும். இல்லையெனில், **Sorry, you are too slow!** என்ற வெளியீடு கிடைக்கும். வழக்கம் போல் எழுதி இயக்கிப் பார்க்கவும். பயனர் செய்யும் தட்டச்சு திரையில் தோன்ற வேண்டாமெனில், **read -s** கட்டளை கொடுக்கவும். எடுத்துக்காட்டாக , பயனர் தமது கடவுச்சொல்லை தட்டச்சு செய்யும் பொழுது அது திரையில் தோன்றாமலிருக்க இதனைப் ப்யன்படுத்தலாம்.

நிரல் **21:**

```
#!/bin/bash

number=0
echo -n "Enter a number > "
read number

echo "Number is $number"
if [ $((number % 2)) -eq 0 ]; then
echo "Number is even"
else
echo "Number is odd"
fi
```

இந்நிரல் புரிந்து கொள்வதற்கும், அமைப்பதற்கும் மிக எளிது. ஓர் எண் ஒற்றைப் படை எண்ணா அல்லது இரட்டைப் படை எண்ணா என்பதை எளிதாக அறிய உதவும் ஒரு நிரல். வழக்கம் போல் எழுதி இயக்கிப்பார்க்கவும்.

நிரல் **22:**

```
#!/bin/bash

echo -n "Enter a number between 1 and 3 inclusive > "
read character
if [ "$character" = "1" ]; then
   echo "You entered one."
else
   if [ "$character" = "2" ]; then
```

```
        echo "You entered two."
    else
        if [ "$character" = "3" ]; then
            echo "You entered three."
        else
            echo "You did not enter a number"
            echo "between 1 and 3."
        fi
    fi
fi
```

இந்நிரல் ஒன்றிலிருந்து மூன்றிற்குள் (மூன்று மற்றும் ஒன்று உட்பட) ஏதேனும் ஒரு எண்ணை உள்ளீடாகக் கொடுத்து அது என்ன என்பதை விளக்குவதாக அமைக்கப்பட்டுள்ளது. இது இருந்தால் அடுக்குக்கட்டளையை (nested if statement) விளக்குவதற்காக கொடுக்கப்பட்டுள்ளது. இதே போன்று ஒரு நிரலமைவைப் பயன்படுத்தி, கொடுக்கப்பட்ட மூன்று எண்களில் பெரியது, சிறியது என்ன (find the biggest or smallest of given three numbers or find the second smallest and second largest numbers) என்பன போன்ற நிரல்களைச் செய்து பார்க்கவும்.

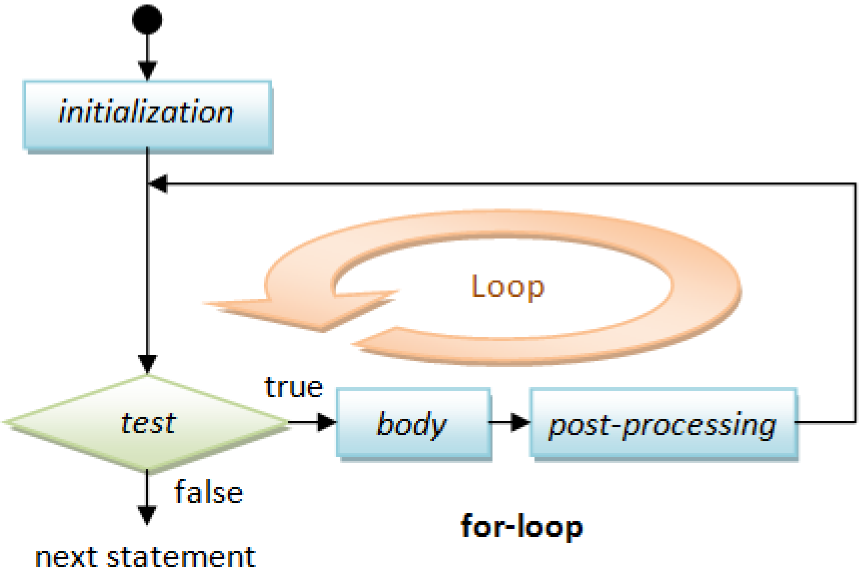
நிரல் **23:**

```
#!/bin/sh
# This is some secure program that uses security.
VALID_PASSWORD="secret" #this is our password.
echo "Please enter the password:"
read PASSWORD
if [ "$PASSWORD" == "$VALID_PASSWORD" ]; then
    echo "You have access!"
else
    echo "ACCESS DENIED!"
fi
```

இந்நிரல் கொடுக்கப்பட்ட கடவுச்சொல் **(password)** சரியானதா இல்லையா என்பதை அறியவுதவும் வண்ணம் அமைக்கப்பட்டுள்ளது. இதை உரை ஒப்பீட்டு நிரலாகவும்

(string comparison script) கூறலாம். இதன் மூலம் பல்வேறு உரை ஒப்பீட்டு செயல்கள் செய்து பார்க்கலாம்.

ஆகக் கட்டளையின் செயல்முறையினை பின்வரும் படமானது நன்கு எடுத்துரைக்கிறது.



முதலில் ஒரு மாறியானது தொடக்க மதிப்பிருத்தல் **(initialization)** செய்யப்படுகிறது. பிறகு குறிப்பிட்ட மாறியானது சோதனை **(testing the condition)** செய்யப்பட்டு, சரியெனில் குறிப்பிட்ட சுழற்சியின் உடலானது **(body of the loop)** ஒருமுறை செய்யப்படுகிறது. தவறெனில் சுழற்சி நிறுத்தப்பட்டு , அடுத்த கட்டளை செயல்படுத்தப்படுகிறது. சோதனையின் விடை சரியாக இருக்கும் வரை குறிப்பிட்ட கட்டளை அல்லது கட்டளைகள் தொடர்ந்து திரும்பத் திரும்ப செய்யப்பட்டுக் கொண்டே இருக்கும் . ஆகக் கட்டளையின் பொது அமைவானது **(General syntax)** கீழ்க்காணுமாறு அமைகிறது.

```
for var in word1 word2 ... wordN
do
  Statement(s) to be executed for every word.
done
```

பின்வரும் இரண்டு நிரல்களைச் செய்து குறிப்பிட்ட வெளியீடுகள் கிடைக்கின்றனவா எனச் சரிபார்க்கவும்.

நிரல் **24:**

```
#!/bin/bash
for var in 0 1 2 3 4 5 6 7 8 9
```

```
do
   echo $var
done
```

[illegible]:

```
0
1
2
3
4
5
6
7
8
9
```

[illegible] 25:

```
#!/bin/sh
for FILE in $HOME/.bash*
do
   echo $FILE
done
```

[illegible]:

```
/root/.bash_history
/root/.bash_logout
/root/.bash_profile
/root/.bashrc
```

[illegible] (more for loop examples:)

[illegible] 26:

```
#!/bin/bash
for X in red green blue
do
        echo $X
done
```

இங்கு **X** என்ற மாறியில் மூன்று மதிப்புகள் இருத்தப்பட்டு மீண்டும் மீண்டும் எடுக்கப்படுகின்றது . நிரலின் செய்து வெளியீடு காணவும்.

நிரல் **27:**

```
#!/bin/bash
colour1="red"
colour2="light blue"
colour3="dark green"
for X in "$colour1" $colour2" $colour3"
do
    echo $X
done
```

இங்கு **X** என்ற மாறியில் மேலும் மூன்று மாறிகள் இருத்தப்பட்டு, அவை திரும்ப அழைக்கப்படுகின்றது.

நிரல் **28:**

```
#!/bin/bash
for X in *.html
do
        grep -L '<UL>' "$X"
done
```

இந்த நிரல் குறிப்பிட்ட ஒரு அடைவுக்குள் இருக்கும் அனைத்து எச்டிஎம்எல் கோப்புகளிலும் **<UL>** என்ற உரை இருக்கிறதா எனப்பார்த்து, அவற்றை மட்டும் திரையில் வெளியிடும். ஒரு அடைவினை உருவாக்கி, அவற்றில் சில எச்டிஎம்எல் கோப்புகளை உருவாக்கி விட்டு, அவற்றில் **UL** உள்ளிட்ட சில வரிகளைத் தட்டச்சு செய்து, பின்பு இந்நிரலை இயக்கிச் சரிபார்க்கவும்.

ஆகக் கட்டளையின் வேறு சில எடுத்துக்காட்டுகளை தொடர்ந்து காணலாம் .

கலைச்சொற்கள்:

நேரிய அறிக்கை - **Proper statements**

சுணக்கம் - **slow down**

சுழற்சியிற் செயற்பட - **running in a loop**

தேவைச் செயல்கள் திரும்பச் செய்யும் **- while condition is right, do the statements again and again**

ஆகக் கட்டளை **- for loop statement**

சுழற்சியின் உடல் – **body of the loop**

மாறி – **variable**

அடைவு – **directory or folder**

உரை – **string or text**

சுழற்சி முறை **- loop**

(கற்போம்...)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட்– 9

சுழற்சி அல்லது ஆகக்கட்டளை **(for loop)** தொடர்ச்சி

சிமொழியில் உள்ள சீரிய அமைவு
அடிபிற ழாமல் இம்மிபி சகாமல்
குறுநிரல் ஆகக் கட்டளைக் குமுண்டாம்
வளைவுக் கட்டளை வேலை செய்யவே.
நிரற்பா - **9**

நிரற்பாவிளக்கம்:

சிமொழியில் நாம் பயன்படுத்தும் அதே பொது அமைவுடன், குறுநிரலிலும் நாம் ஆகக் கட்டளை வரிகளை அமைக்கலாம். சிமொழியில் அமைந்துள்ள அதே வகையான அமைப்பு அமைக்கப்படுவதால் அடிபிறழாமல், இம்மிபிசகாமல் என்று கொடுக்கப்பட்டுள்ளது. ஆகக் கட்டளை இங்கு வளைவுக்கட்டளை என்று கொடுக்கப்பட்டுள்ளது.

ஆகக் கட்டளையின் வேறு சில எடுத்துக்காட்டுகளை தொடர்ந்து காண்பதற்கு முன்னால் நாம் சிமொழியில் வளைவுக்கட்டளையின் பொதுஅமைவினைக் காணலாம்.

```
for ((i=0;i>=10;i++))
{
 Set of statements
}
```

நாம் முன்பு வேறு வகையான குறுநிரல்களைக் கண்டோம். இப்பொழுது சிமொழியின் பொது அமைவினைக் கொண்ட குறுநிரல்களைக் காணலாம்.

நிரல் **29:**

```
#!/bin/bash
# Random number generation using C language syntax
for (( i=1; i <= 5; i++ ))
```

```
do
 echo "Random number $i: $RANDOM"
done
```

இந்நிரலில் சிமொழியிலுள்ள அதே வகையான பொது அமைவு உள்ளது. { } என்னும் குறியீடுகளுக்கு (curly braces) மாற்றாக do, done என்னும் சொற்கள் உள்ளன. 5 முறை இந்த சுழற்சியானது இயக்கப்படுகிறது. வெளியீடு பின்வருமாறு அமைகிறது.

வெளியீடு**:**

```
$ ./niral29.sh
Random number 1: 23320
Random number 2: 5070
Random number 3: 15202
Random number 4: 23861
Random number 5: 23435
```

நிரல் **30:**

```
#!/bin/bash
# This program explains about infinite loop in shell script
i=1;
for (( ; ; ))
do
   sleep $i
   echo "Number: $((i++))"
done
```

இது ஒரு முடிவுறா சுழற்சி வளைவுக்கட்டளையாக அமைகிறது. ஆகக் கட்டளைக்குள் இருக்கும் கட்டளை வரிகள் திரும்பத்திரும்ப இயக்கப்பட்டு வெளியீடானது கிடைக்கிறது. இது சிமொழியில் உள்ள அதே வகையான அமைவாகும். இங்கு மாறியானது ஒன்று கூட்டப்பட்டு வருவதால், வளைவுக்கட்டளை எத்தனை முறை இயக்கப்பட்டது என்பதைக் காண இயலுமாறு வெளியீடு அமைகிறது .

வெளியீடு**:**

```
$ ./niral30.sh
```

```
Number: 1
Number: 2
Number: 3
```

நிரல் **31:**

```
#!/bin/bash
#for loop using comma

for ((i=1, j=10; i <= 5 ; i++, j=j+5))
do
 echo "Number $i: $j"
done
```

இந்நிரலிலும் சிமொழியில் உள்ள அதே வகையான அமைவு எடுத்தாளப்பட்டுள்ளது. இங்கு i மற்றும் j என்று இரண்டு வகையான மாறிகள் கையாளப்படுகின்றன . இரண்டும் தனித்தனியாக ஒன்றுடன் கூட்டப்படுகின்றன அதுவும் ஒரே ஒரு கட்டளை வரியில். வெளியீடானது பின்வருமாறு அமைகிறது. இங்கு இரண்டு மாறிகள் எடுத்துக்காட்டிற்காக கொடுக்கப்பட்டுள்ளன . இதே போல் பல மாறிகளை ஒரே கட்டளைவரியில் பயன்படுத்தி கணக்கீடுகளைச் செய்து வெளியீடுகளை எளிமையாகக் காணலாம்.

வெளியீடு**:**

```
$ ./niral31.sh
Number 1: 10
Number 2: 15
Number 3: 20
Number 4: 25
Number 5: 30
```

நிரல் **32:**

```
#!/bin/bash
#! For loop with range in numbers
```

```
for num in {1..10}
do
 echo "Number: $num"
done
```

இங்கு சிமொழியின் பொது அமைவு கையாளப்படவில்லை. ஆகக் கட்டளை கொண்டு ஒன்று முதல் பத்து வரை அச்சிட கட்டளை கொடுக்கப்பட்டுள்ளது .

வெளியீடு**:**

```
$ ./niral32.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 9
Number: 10
```

இதே நிரலை சிமொழி அமைவுடன் பின்வருமாறு அமைக்கலாம்**.**

நிரல் **33:**

```
#!/bin/bash
#for loop with with C language syntax
for ((i=1;i<=10;i++))
do
echo “Number: $i”
done
```

நிரல் **34:**

```
#!/bin/bash
# Range of numbers with increments after “in” keyword
```

```
for num in {1..10..2}
do
 echo "Number: $num"
done
```

[illegible]:

```
$ ./niral34.sh
Number: 1
Number: 3
Number: 5
Number: 7
Number: 9
```

[illegible] [illegible] [illegible], [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

[illegible] 35:

```
#!/bin/bash
#Niral 34 with C Language syntax
for ((i=1;i<=10;i=i+2))
do
     echo ”Number: $i”
done
```

[illegible] 36:

```
#!/bin/bash
# fileinfo.sh  Fileinfo: operating on a file list contained in a variable
FILES="/usr/sbin/accept
/usr/sbin/pwck
/usr/sbin/chroot
/usr/bin/fakefile
```

```
/sbin/badblocks
/sbin/ypbind"     # List of files you are curious about.
                  # Threw in a dummy file, /usr/bin/fakefile.

echo

for file in $FILES
do
  if [ ! -e "$file" ]       # Check if file exists.
  then
    echo "$file does not exist."; echo
    continue                # On to next.
   fi

  ls -l $file | awk '{ print $9 "         file size: " $5 }'  # Print 2 fields.
  whatis `basename $file`   # File info.
  # Note that the whatis database needs to have been set up for this
to work.
  # To do this, as root run /usr/bin/makewhatis.
  echo
done

exit 0
```

[illegible] (file path) [illegible], [illegible], [illegible]. [illegible].

[illegible]:

[illegible] – **general syntax**

வளைவுக்கட்டளை – for loop
குறுநிரல் – shell script
முடிவுறா சுழற்சி வளைவுக்கட்டளை – infinite for loop
மாறி - variable
(கற்போம்...)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -77- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -10 வளைவுக் கட்டளையின் வேறு வகைகள்

பொழுதெலாம் கட்டளை நுழைவு அடக்கம்
வரையிலும் கட்டளை வெளிப்புற அடக்கம் என்பன போன்று இன்னும் உண்டாம்
வளைவுக் கட்டளை வகைகள் இரண்டே.
- நிரற்பா 10

நிரற்பா விளக்கம் :

வளைவுக் கட்டளையானது இன்னும் இரண்டு வகைகளில் உயர் நிலை மொழிகளில் கையாளப்படுகிறது. ஒன்று ஒரு கட்டுப்பாட்டினை (condition) ஏற்றுக் கொண்டு அது சரியெனில் அதற்குரிய கட்டளைகளை இயக்குதல். இது பொழுதெலாம் கட்டளை கொண்டு செயல்படும் வளைவுக் கட்டளையாகும் (while loop – entry controlled loop). இது நுழைவு அடக்கம் அல்லது நுழைவுக் கட்டுப்பாடு என்ற கருத்துருவில் நிகழ்கிறது. இரண்டாவது, சில கட்டளை வரிகளை இயக்கிவிட்டு அதன்பிறகு குறிப்பிட்ட கட்டுப்பாட்டினை சரிபார்க்கும் கருத்துரு. இது வரையிலும் கட்டளை வெளிப்புற அடக்கம் (while loop - exit controlled loop) என்று அழைக்கப்படுகிறது. பின்வரும் படம் இதனை நன்கு விளக்குவதாக அமைந்துள்ளது. இதனை முன் சோதனை வளைவுக் கட்டளை (pretest loop) பின் சோதனை வளைவுக் கட்டளை (post test loop) என்றும் அழைக்கலாம்.
வரையிலும் கட்டளைக்கு எதிராக பொழுதெலாம் கட்டளை (whle loop vs until loop) மூன்று படிகளைக் கொண்டுள்ளது.
1. வரையிலும் கட்டளை சுழியல்லாத நிலை (non-zero status) வரை செய்யப்படுகிறது.
2. பொழுதெலாம் கட்டளை சுழி நிலை (zero status) வரை செய்யப்படுகிறது.
3. வரையிலும் கட்டளை எப்பொழுதுமே ஒரு முறை தனது கட்டளை வரிகளை இயக்கிவிடுகிறது.

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -78- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

TEST CONTROL VARIABLE
F
T
STATEMENT TO BE REPEATED
PRETEST LOOP

STATEMENT TO BE REPEATED
TEST CONTROL VARIABLE
F
T
POSTTEST LOOP

[illegible] **(some more for statement examples):**

[illegible] **37:**

```
#!/bin/bash
# print the user names
i=1
for username in `awk -F: '{print $1}' /etc/passwd`
do
 echo "Username $((i++)) : $username"
done
```

[illegible]**:**

[illegible] **(system)** [illegible] **(usernames)** [illegible]**.** **/etc/passwd** [illegible]**.** [illegible] **awk** [illegible]**.**

வெளியீடு**:**

```
# ./niral36.sh
Username 1 : prasanna
Username 2 : arivu
Username 3 : nedilan
Username 4 : porrko
..
```

நிரல் **37:**

```
#!/bin/bash
i=1
cd ~
for item in *
do
 echo "Item $((i++)) : $item"
done
```

நிரல் விளக்கம்**:**

இங்கு குறிப்பிட்ட அடைவிலுள்ள அனைத்து வகையான கோப்புக்களையும் காட்டும் வகையில் நிரல் அமைக்கப்பட்டுள்ளது**. cd ~** என்பது ஒரு குறிப்பிட்ட பயனரின் வீட்டு அல்லது முகப்பு அடைவினைக் குறிக்கிறது**.**

வெளியீடு**:**

```
# ./niral37.sh
Item 1 : positional-parameters.sh
Item 2 : backup.sh
Item 3 : emp-report.awk
Item 4 : item-list.sed
Item 5 : employee.db
Item 8 : storage
Item 9 : downloads
```

நிரல் **38:**

```
#!/bin/bash
i=1
for file in /etc/[abcd]*.conf
```

```
do
 echo "File $((i++)) : $file"
done
```

நிரல் விளக்கம்:

இந்த நிரலில் குறிப்பிட்ட எழுத்துக்களில் ஏதேனும் ஒன்றைக் கொண்டு தொடங்கக்கூடிய **(a,b,c,d** ஆகியவற்றில் ஏதேனும் ஒரு எழுத்தில் தொடங்கக்கூடிய**)** கோப்புகள் மட்டும் வெளியிடப்படும் வண்ணம் அமைக்கப்பட்டுள்ளது. மற்ற கோப்புகள் நிரலின் வெளியீட்டில் காட்டப்படாது.

வெளியீடு:

```
# ./niral38.sh
File 1 : /etc/asound.conf
File 2 : /etc/autofs_ldap_auth.conf
File 3 : /etc/cas.conf
File 4 : /etc/cgconfig.conf
File 5 : /etc/cgrules.conf
File 6 : /etc/dracut.conf
```

நிரல் **39:**

```
#!/bin/bash
# Script for multiplication table for 5
#
if [ $# -eq 0 ]
then
echo "Error - Number missing form command line argument"
echo "Syntax : $0 number"
echo "Use to print multiplication table for given number"
exit 1
fi
n=5
for i in 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
done
```

நிரல் விளக்கம்:
மேலேயுள்ள நிரல், ஐந்தாம் வாய்ப்பாட்டினை அமைக்க உதவுகிறது. இங்கு n=5 என்ற கட்டளை வரியில் வேறொரு எண்ணை அமைத்தால், வேறு வகையான வாய்ப்பாட்டினை வெளியீடாகக் காணலாம்.
வெளியீடு:
./niral39.sh
5 * 1 = 5
5 * 2 = 10

...
..

5 * 10 = 50
இதே நிரலினை வேறு வகையான ஆகக் கட்டளை கொண்டு (for loop another format) எழுதி இயக்கிப்பார்த்துப் பயிற்சி எடுக்கவும்.
பொழுதெலாம் கட்டளை (while loop) எடுத்துக்காட்டுக்களை பிறகு காணலாம்.

கலைச்சொற்கள் :

வரையிலும் கட்டளை – until loop
பொழுதெலாம் கட்டளை – while loop
வெளிப்புற அடக்கம் – exit controlled
நுழைவு அடக்கம் – entry controlled
வளைவுக் கட்டளை – loop statements
கருத்துரு - concept
சுழி நிலை – zero status
சுழியல்லாத நிலை – non-zero status
பொறி – system
பயனர் பெயர் – username
வீட்டு அல்லது முகப்பு அடைவு – Home folder or home directory (கற்போம்...)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -82- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

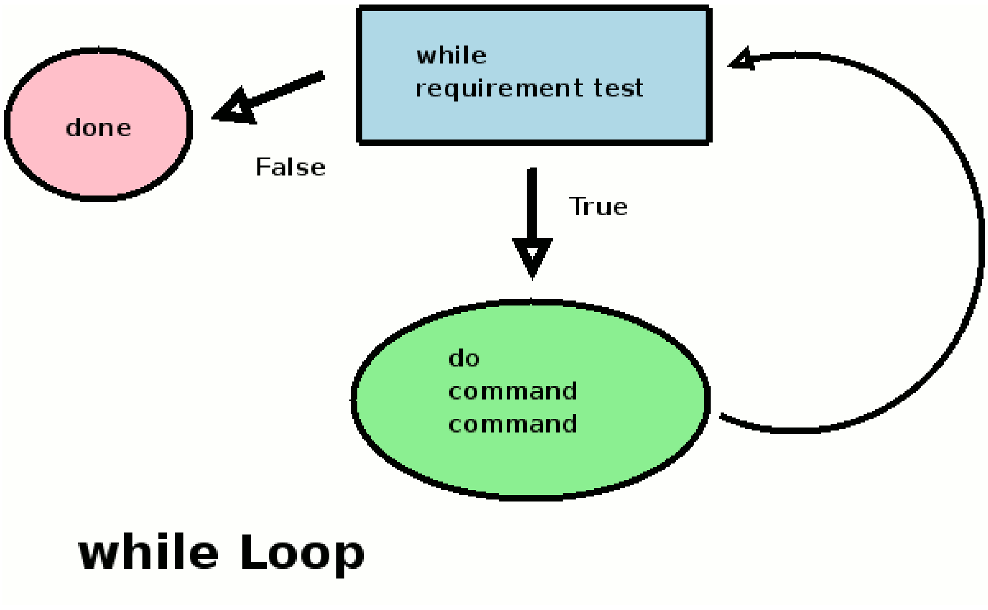
செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -11

பொழுதெலாம் கட்டளை அல்லது **while loop – entry controlled loop**

நுழைவு அடக்கம் நுட்பமாய்க் கொண்ட
பொழுதெலாம் கட்டளை புத்துணர் வோடு
தன்னுடல் பூண்ட தவறா வரிகளை
செய்யுமாம் சுழிநிலை வருகின்ற வரையிலே.
- நிரற்பா **11**

நிரற்பா விளக்கம்:

வளைவுக்கட்டளையின் நுழைவிலேயே கொடுக்கப்பட்டுள்ள, கட்டுப்பாடானது நிறைவடைகிறதா **(whether the given condition is satisfied or not)** எனச் சரிபார்த்து , சரியெனில் தனது உடலினை **(body of the loop)** திரும்பத்திரும்ப செய்யும். பொதுவாக, கட்டுப்பாடானது சுழிநிலையாகவே இருக்கும். இப்பாவில், சுழிநிலை வருகின்ற வரையிலே என்ற வரி, குறிப்பிட்ட கட்டுப்பாடானது நிறைவடைவதையே குறிக்கிறது. கட்டுப்பாடானது நிரலினைப் பொறுத்து எந்த வகையானதாகவும் இருக்கலாம். இங்கு சுழி என்பது சுழியத்தை **(zero)** மட்டும் குறிக்காமல், இல்லை **(condition is false)** என்பதையும் குறிக்கிறது.



[illegible].

[illegible] 40:

```
#!/bin/bash
#niral40
#chessboard
for (( i = 1; i <= 9; i++ )) ### Outer for loop ###
do
   for (( j = 1 ; j <= 9; j++ )) ### Inner for loop ###
   do
        tot=`expr $i + $j`
        tmp=`expr $tot % 2`
        if [ $tmp -eq 0 ]; then
            echo -e -n "\033[47m "
        else
            echo -e -n "\033[40m "
        fi
  done
 echo -e -n "\033[40m" #### set back background colour to black
 echo "" #### print the new line ###
done
```

[illegible]:

[illegible] (chess board) [illegible].

[illegible]	[illegible]
for ((i = 1; i <= 9; i++)) do	Begin the outer loop which runs 9 times., and the outer loop terminates when the value of i exceeds 9
for ((j = 1 ; j <= 9; j++)) do	Begins the inner loop, for each value of i the inner loop is cycled through 9 times, with the variable j taking values from 1 to 9. The inner for loop terminates when the value of j

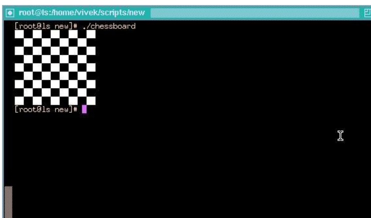
	exceeds 9.
tot=`expr $i + $j` tmp=`expr $tot % 2`	See for even and odd number positions using these statements.

if [$tmp -eq 0]; then
   echo -e -n "\033[47m "
else
   echo -e -n "\033[40m "
fi
	If even number position print the white colour block (using echo -e -n "\033[47m "statement); otherwise for odd position print the black colour box (using echo -e -n "\033[40m " statement). These statements are responsible to print entire chess board on screen with alternate colours.

done	End of inner loop
echo -e -n "\033[40m"	Make sure its black background as we always have on our terminals.
echo ""	Print the blank line
done	End of outer loop and shell scripts get terminated by printing the chess board.

□□□□□□□□:

#./niral40.sh



□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□ □□□ (Syntaxes of while loop):

□□□□□□ □□□□□□□□ □□□□□□□□□□□ □□□□□□□□□ while □□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□□□. □□□□□□□□ □□□□□□□□□□□.

Ksh shell syntax:

```
while [[ condition ]] ; do
    command1
    command1
    commandN
```

```
done
csh syntax:
while command
do
   Statement(s) to be executed if command is true
done
Bash syntax:
while [ condition ]
do
     command1
     command2
     commandN
done
```

இனி **(while)** பொழுதெலாம் கட்டளையின் எடுத்துக்காட்டுக்களை காணலாம் **.**

நிரல் **41:**

```
#!/bin/sh
#niral 41
a=0
while [ $a -lt 10 ]
do
   echo $a
   a=`expr $a + 1`
done
```

நிரல் விளக்கம்**:**

இனி **a** என்ற மாறியானது**,** பத்து என்ற எண் வரும் வரை குறிப்பிட்ட கட்டளையினைத் திரும்பத்திரும்ப செய்து பின்வரும் வெளியீட்டினை அளிக்கிறது**. lt** என்பது **less than** ஐக் குறிப்பதால் **10** என்ற எண்ணை விட்டுவிட்டு ஒன்பது வரை பதிப்பிக்கிறது **(printing).** இந்தக் கட்டளையினை நாம் ஆகக் கட்டளைக்கு **(an alternate for for command)** மாற்றாகப் பயன்படுத்தலாம்**.**

வெளியீடு**:**

```
#./niral41.sh
0
```

1

2

3

4

5

6

7

8

9

நிரல் **42:**

```
#!/bin/bash
#niral 42
c=1
while [ $c -le 5 ]
do
    echo "Welcome $c times"
    (( c++ ))
done
```

நிரல் விளக்கம்**:**

இந்த நிரலிலும் மேற்குறிய நிரல் போலவே குறிப்பிட்ட கட்டுப்பாடு நிறைவடையும் **(satisfying the given condition)** வரை கட்டளைகள் செயல்படுத்தப்பட்டு வெளியீடானது பின்வருமாறு அமைகிறது **.**

வெளியீடு**:**

```
#./niral42.sh
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
```

நிரல் **43:**

```
#!/bin/bash
#niral 43
```

```
# This generates a file every 5 minutes
while true; do
touch pic-`date +%s`.jpg
sleep 300
done
```

நிரல் விளக்கம்:

இங்கு **while** கட்டளையானது, ஒரு முடிவுறா வளைவுக்கட்டளையைக் கொடுக்கிறது. அதாவது அது எப்பொழுதுமே **true** ஆக இருக்கிறது. **touch** கட்டளை ஒரு குறிப்பிட்ட கோப்பினை உருவாக்குவதற்காக கொடுக்கப்பட்டுள்ளது. **sleep 300** கட்டளை ஐந்து மணித்துளிகள் (அல்லது **300** நொடிகள்) பொறியினை ஒன்றும் செய்யவிடாமல் தூங்க வைக்கிறது.

வெளியீடு:

#./niral43.sh

இந்த நிரலானது, ஐந்து மணித்துளிகளுக்கு ஒருமுறை ஒரு படக்கோப்பினை குறிப்பிட்ட அடைவிற்குள் உருவாக்குகிறது.

கலைச்சொற்கள்:

நுழைவு அடக்கம் – **entry controlled**

பொழுதெலாம் கட்டளை – **while loop**

தன்னுடல் பூண்ட – **body of the loop**

சுழிநிலை – **zero status, condition is false**

சுழியம், இல்லை, இன்மை – **zero**

சதுரங்கப்பலகை – **chess board**

(கற்போம்...)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -12

வரையிலும் கட்டளை அல்லது **until loop – exit controlled loop**

வெளிப்புற அடக்கம் வழியில் காணும்
வரையிலும் கட்டளை விரும்பி மெய்பொய்
காணாமல் சுழியறக் குறைந்த தொருமுறை
யேனும் செய்யுமாம் வேண்டு மென்றே.
- நிரற்பா **12**

நிரற்பா விளக்கம்:

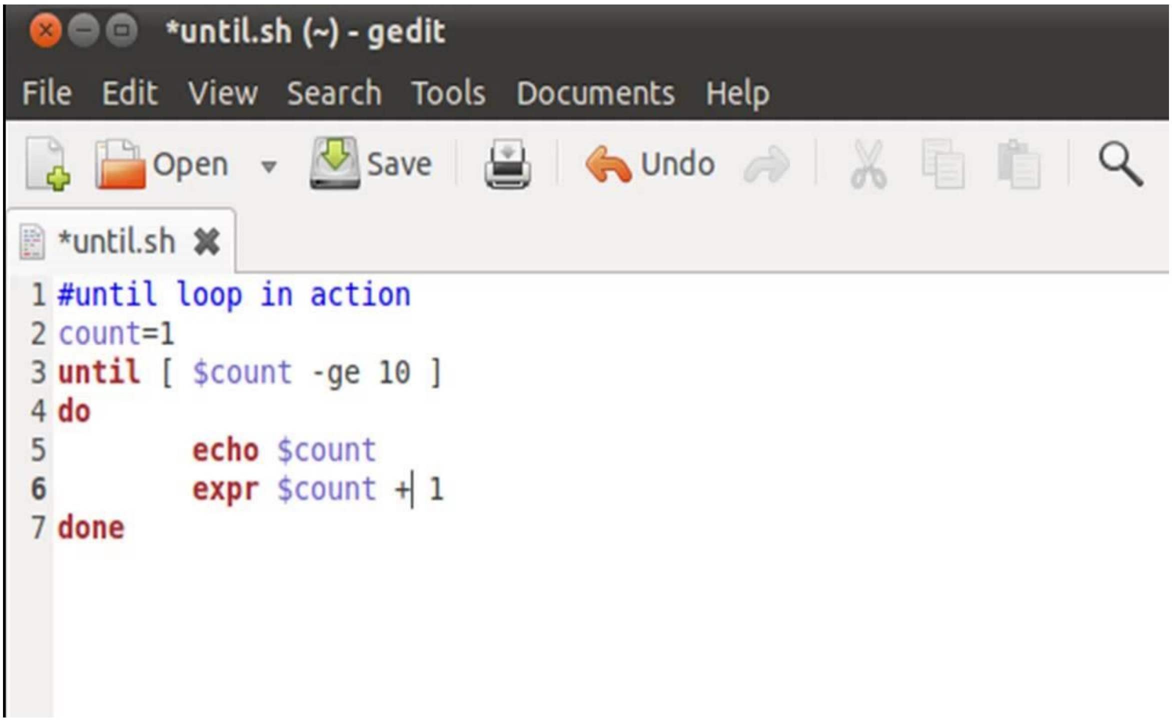
வரையிலும் கட்டளையானது தனக்குக் கொடுக்கப்பட்ட கட்டுப்பாடானது சரி அல்லது தவறு என்று எதையும் பாராமல், ஒரு முறையாவது தனக்குக் கொடுக்கப்பட்டுள்ள கட்டளைகள் அனைத்தையும் செய்து முடிக்கும். இங்கு சுழியற என்பது **non-zero value** ஐக் குறிக்கிறது .இது அனைத்து வகையான உயர் நிலை மொழிகளிலும் உண்டு என்றாலும், இதைப் பயன்படுத்தும் நிரலர்கள் குறைவே.

பொதுவமைவு **(General Syntax):**

```
until expression
    do
      commands #body of the loop
    done
```

மேலே குறிப்பிடப்பட்டுள்ள பொதுவமைவு கொண்ட இக்கட்டளை சற்றே அரிதாக நிரலர்கள், மற்றும் மென்பொறிஞர்களால் கையாளப்படுகிறது. பெரும்பாலும், பொழுதெலாம் கட்டளை **(while loop command)** கொண்டே இது போன்ற செயல்கள் செய்யப்படுகின்றன. இதுவும் ஏறத்தாழ பொழுதெலாம் கட்டளை போன்றதே. இதில்

கட்டளை உண்மை எனினும் அல்லது பொய் எனினும் ஒரு முறை வளைவுக் கட்டளையானது செயல்படும்.



நிரல் **44:**

```
# cat monitor.sh
#!/bin/bash
#niral 44
file=/tmp/logfile
until [ $(ls -l $file | awk '{print $5}') -gt 2000 ]
      do
         echo "Sleeping for next 5 seconds"
         sleep 5
      done
date=`date +%s`
cp $file "$file-"$date.bak
```

நிரல் விளக்கம்:

இந்நிரலில், குறிப்பிட்ட **log** கோப்பு உருவாக்கப்படுகிறது. அது **2000** பைட்டுகள் ஆன பிறகு அதை காப்புப்படி எடுத்து வைக்கிறது. அது **.bak** என்னும் பின்னொட்டுடன் இருக்கிறது. இங்கு வரையிலும் கட்டளையானது தொடர்ந்து

செய்யப்படுகிறது. இந்த கோப்புகள் அனைத்தும் **/tmp** என்ற அடைவுக்குள் உருவாக்கப்படுகிறது.

வெளியீடு:

```
# ./monitor.sh
Sleeping for next 5 seconds
Sleeping for next 5 seconds

# ls -l /tmp/logfile*
-rw-r--r-- 1 sss sss     2010 Jun 24 12:29 logfile
-rw-r--r-- 1 sss sss     2005 Jun 24 16:09 logfile-1277474574.bak
```

நிரல் **45:**

```
# cat mac_wait.sh
#! /bin/bash
#niral  45
read -p "Enter IP Address:" ipadd
echo $ipadd
until ping -c 1 $ipadd
do
	sleep 60;
done
ssh $ipadd
```

நிரல் விளக்கம்:

இக்குறுநிரலானது, **ssh** கட்டளையைப் பயன்படுத்தி, மற்றொரு பொறியினை தொலை அணுகல் மூலம் அணுகும் **(remote access)** பொழுது முதலில் அது இணைப்பொலிக் கூவல் **(ping command)** மூலம் தொடர்பில் இருக்கிறதா என சரி பார்க்கப்படுகிறது . பின்பு அறுபது நொடிகள் தூங்குகிறது. பிறகு மீண்டும் கூவல் செய்கிறது. விடைகள் கிடைத்தால் மட்டும் தொலை அணுகலுக்கான கட்டளையினைச் செயல்படுத்துகிறது. இணைப்பொலிக் கூவல் விடைதறாவிடில் தொலை அணுகல் செய்து நேரத்தை வீணாக்காது. பின்வரும் எடுத்துக்காட்டான வெளியீட்டினைக் காணவும்.

[illegible]:

```
#./mac_wait.sh
Enter IP Address:192.143.2.10
PING 192.143.2.10 (192.143.2.10) 56(84) bytes of data.

--- 192.143.2.10 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

PING 192.143.2.10 (192.143.2.10) 56(84) bytes of data.
64 bytes from 192.143.2.10: icmp_seq=1 ttl=64 time=0.059 ms

--- 192.143.2.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.059/0.059/0.059/0.000 ms
The authenticity of host '192.143.2.10 (192.143.2.10)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
```

[illegible] 46:

```
#!/bin/bash
#niral 46
number=0
until [ $number -ge 10 ]; do
    echo "Number = $number"
    number=$((number + 1))
done
```

[illegible]:

[illegible].

[illegible]:

[illegible].

[illegible] 47:

```
#!/bin/bash
# This script copies files from my homedirectory into the webserver
directory.
# A new directory is created every hour.
# If the pics are taking up too much space, the oldest are removed.

while true; do
    DISKFUL=$(df -h $WEBDIR | grep -v File | awk '{print $5 }' | cut -d
"%" -f1 -)

    until [ $DISKFUL -ge "90" ]; do

        DATE=`date +%Y%m%d`
        HOUR=`date +%H`
        mkdir $WEBDIR/"$DATE"

        while [ $HOUR -ne "00" ]; do
                DESTDIR=$WEBDIR/"$DATE"/"$HOUR"
                mkdir "$DESTDIR"
                mv $PICDIR/*.jpg "$DESTDIR"/
                sleep 3600
                HOUR=`date +%H`
        done

    DISKFULL=$(df -h $WEBDIR | grep -v File | awk '{ print $5 }' | cut -d
"%" -f1 -)
    done

    TOREMOVE=$(find $WEBDIR -type d -a -mtime +30)
    for i in $TOREMOVE; do
        rm -rf "$i";
    done
```

done
நிரல் விளக்கம்:
இந்நிரலில் குறிப்பிட்ட ஒரு முகப்பு அடைவானது (home directory files) அதிலுள்ள கோப்புக்களை ஒரு வலை வழங்கியில் படியெடுத்து வைக்கப்படுகிறது. ஒரு புதிய அடைவானது (new directory) ஒவ்வொரு மணி நேரத்திற்கும் உருவாக்கப்படுகிறது.
படங்களானது அதிகமாக இருப்பின், பழைய படங்கள் நீக்கப்பட்டு விடும். வெளியீடு:
மேலே குறித்த நிரல் விளக்கங்களைக் கொண்டு நிரலினை செய்து பார்த்து விடையறிய முயலுக.

கலைச்சொற்கள் :

வெளிப்புற அடக்கம் – exit controlled
வரையிலும் கட்டளை – until loop
மெய்பொய் – Boolean value (it may be true or false)
சுழியற – non zero value
குறைந்த தொருமுறை – at least once
வலை வழங்கி – web server
படியெடுத்தல் – backup
இணைப்பொலிக் கூவல் – connection ping command
கூவல் விடைகள் – ping replies
தன்புரிதல் – self explanatory
(கற்போம்…)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -94- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் **-13.**

தேர்வுக் கட்டளை அல்லது தேர்வாணை **(case statement)**

தன்னுள் ஏற்ற தீர்வு வரிகளை
வகைக்கொரு கட்டளை வண்ணம் பிரித்து
மற்றவை வகைமை முறையில் நோக்கித்
திறம்படச் செய்யும் தேர்வா ணையே.
நிரற்பா **-13**

நிரற்பா விளக்கம்:

இங்கு தேர்வு ஆணை எனப்படுவது **case** கட்டளையினைக் குறிப்பதாகும்.
Case கட்டளை எல்லா உயர் நிலை மொழிகளைப் போன்றும், ஷெல் ஸ்கிரிப்டிலும் , வகைக்கொரு வண்ணமாகச் செயல்பட்டு அதற்குரிய ஒன்று அல்லது அதற்கு மேற்பட்ட கட்டளைகளை இயக்கித் தீர்வினைத் தருகிறது.
இந்தக் கட்டளையின் பொதுவமைவானது கீழே கொடுக்கப்பட்டுள்ளது. இந்தக் கட்டளையினை **if else** ஏணிக்கு **(else if ladder)** மாற்றாகப் பயன்படுத்தலாம்.
குறிப்பு **:** இங்கு **case** எனத் தொடங்கும் தேர்வுக் கட்டளை **esac** என்று முடிதல் வேண்டும்.
பொது அமைவு **(General syntax):**

```
case word in
  pattern1)
     Statement(s) to be executed if pattern1 matches
     ;;
  pattern2)
     Statement(s) to be executed if pattern2 matches
     ;;
  pattern3)
     Statement(s) to be executed if pattern3 matches
```

```
    ;;
  *)
    Default statement(s)
esac
```

இங்கு * ஆனது மற்ற எந்த வகைக் கட்டளைகளிலும், இல்லாத வகை **(pattern)** ஏதேனும் வரப்பெற்றால் அதனை இயக்குவதாக அமைகிறது .

நிரல் **48:**

```
#!/bin/sh
#niral 48
FRUIT="kiwi"
case "$FRUIT" in
   "apple") echo "Apple pie is quite tasty."
   ;;
   "banana") echo "I like banana nut bread."
   ;;
   "kiwi") echo "New Zealand is famous for kiwi."
   ;;
esac
```

நிரல் விளக்கம்**:**

இங்கு நிரலானது **kiwi** என்ற வகையீட்டில் வருவதால்**,**கீழ்க்காணுமாறு வெளியீடானது அமைகிறது .

வெளியீடு **(Output):**

New Zealand is famous for kiwi.

நிரல் **49:**

```
# cat filetype.sh
#!/bin/bash
#niral 49
for filename in $(ls)
do
     # Take extension available in a filename
        ext=${filename##*\.}
        case "$ext" in
```

```
        c) echo "$filename : C source file"
           ;;
        o) echo "$filename : Object file"
           ;;
        sh) echo "$filename : Shell script"
            ;;
        txt) echo "$filename : Text file"
             ;;
        *) echo " $filename : Not processed"
           ;;
esac
done
```

[illegible] [illegible]:

[illegible] [illegible] [illegible] (folder) [illegible], [illegible] [illegible] (type of the file and extension) [illegible] [illegible] [illegible]. [illegible], [illegible] [illegible] [illegible], [illegible], [illegible] [illegible], [illegible] [illegible] [illegible] (file without extension or Not processed file) [illegible] [illegible].

[illegible] (Output):

```
# ./filetype.sh
a.c : C source file
b.c : C source file
c1.txt : Text file
fileop.sh : Shell script
obj.o : Object file
text : Not processed
t.o : Object file
```

[illegible] 50:

```
# cat yorno.sh
#niral50.sh
#!/bin/bash
echo -n "Do you agree with this? [yes or no]: "
```

```
read yno
case $yno in

	[yY] | [yY][Ee][Ss] )
		echo "Agreed"
		;;

	[nN] | [n|N][O|o] )
		echo "Not agreed, you can't proceed the installation";
		exit 1
		;;
	*) echo "Invalid input"
	    ;;
esac
```

நிரல் விளக்கம்:

இந்நிரலானது ஒரு பயனர் ஏற்கிறாரா? மறுக்கிறாரா? என்பதைக் கேட்டுப் பெறும் நிரலாக அமைகிறது. **(an user is agreeing or disagreeing)** பார்க்க மிக எளிமையான நிரலாக இருப்பினும், இது கொடுக்கப்படும் உரையினை **(checking the given text)** சரிபார்க்க உதவுகிறது.

வெளியீடு **(Output):**

```
# ./yorno.sh
Do you agree with this? [yes or no]: YES
Agreed
```

நிரல் **51:**

```
#!/bin/sh
# Wedding guest meals

# These variables hold the counters.
NUM_CHICKEN=0
NUM_STEAK=0
ERR_MSG=""
```

```
# This will clear the screen before displaying the menu.
clear

while :
do
    # If error exists, display it
    if [ "$ERR_MSG" != "" ]; then
        echo "Error: $ERR_MSG"
        echo ""
    fi

    # Write out the menu options...
    echo "Chicken: $NUM_CHICKEN"
    echo "Steak: $NUM_STEAK"
    echo ""
    echo "Select an option:"
    echo " * 1: Chicken"
    echo " * 2: Steak"
    echo " * 3: Exit"

    # Clear the error message
    ERR_MSG=""

    # Read the user input
    read SEL

    case $SEL in
        1) NUM_CHICKEN=`expr $NUM_CHICKEN + 1` ;;
        2) NUM_STEAK=`expr $NUM_STEAK + 1` ;;
        3) echo "Bye!"; exit ;;
        *) ERR_MSG="Please enter a valid option!"
    esac
```

 # This will clear the screen so we can redisplay the menu.

clear

done

நிரல் விளக்கம்**:**

இது ஒரு மெனு என்றழைக்கப்படக்கூடிய, பயனர் தேர்ந்தெடுப்பான் **(user menu)** அமைப்பதற்காக உதவும் நிரலாகும். இதில் நாம் ஏற்கனவே பார்த்த**,** பொழுதெலாம் கட்டளை**,** இருந்தால் கட்டளை ஆகியவையும் பயின்று வந்துள்ளன**.** நிரலினை இயக்கிப்பார்த்து விடையறிவும்.

கலைச்சொற்கள்**:**

தேர்வாணை – **case statement**

மற்றவை வகைமை – **default pattern**

வகை – **pattern**

தீர்வு வரிக ள் – **solution or result statements**

பொழுதெலாம் கட்டளை – **while statement**

இருந்தால் கட்டளை – **if statement**

பயனர் தேர்ந்தெடுப்பான் – **user menu**

(கற்போம்**)**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -14.
தேர்வுக் கட்டளை இருந்தால் கட்டளை வேறுபாடுகள் (case statement vs else if ladder)

பற்பல மதிப்புகள் பொறுப்பாய் ஏற்கும்
இருந்தால் கட்டளை இயல்பாய்ச் செய்ய
மதிப்பு ஒன்றை மட்டும் ஏற்கும்
தேர்வுக் கட்டளை திண்ணமாய் விரையுமே.
நிரற்பா -14

நிரற்பா விளக்கம்:

இருந்தால் கட்டளையானது பற்பல மதிப்புகளை ஏற்கும் தன்மை கொண்டதாகும் . மேலும் அது சற்றே மெதுவாகச் செயல்படும் தன்மை கொண்டது. ஆனால் தேர்வுக்கட்டளையானது ஒரே ஒரு மதிப்பை மட்டுமே தன்னகத்தே ஏற்கக்கூடியது . எனவே அது இருந்தால் கட்டளையினைக் காட்டிலும் விரைவாகச் செயற்படக்கூடியது . இது அனைத்து வகையான உயர் நிலை மொழிகளுக்கும் பொருந்தும் .

இருப்பினும், ஒன்றிற்கு மேற்பட்ட மாறிகளைக் கொண்டு தேர்வுக் கட்டளையினை நாம் சுற்றி வளைத்து ஒன்றிற்கு மேற்பட்ட மதிப்புக்களை இருத்தி இயக்கலாம்.

இது தொடர்பாக உள்ள நிரல்களை இப்பொழுது காணலாம். முதலில் இருந்தால் ஏணிக்கட்டளையினைக் **(else if ladder command in shell script)** காணலாம்.

நிரல் **52:**

```
#!/bin/bash
#niral52.sh
#example for elif ladder in shell script
read -p "Enter value of i :" i

if [ $i -eq 5 ]
then
```

```
   echo "Value of i is 5"
elif [ $i -eq 10 ]
then
   echo "Value of i is 10"
elif [ $i -eq 20 ]
then
   echo "Value of i is 20"
elif [ $i -eq 30 ]
then
   echo "Value of i is 30"
else
   echo "Value of i is not equal to 5,10,20 or 30"
fi
```

நிரல் விளக்கம்:

இந்நிரலில் ஒரு மதிப்பானது உள்ளீடாகப் பெறப்படுகிறது . அது ஒவ்வொரு கட்டுப்பாடுகளாகச் சரிபார்த்து உரிய கட்டுப்பாடு நிறைவடைந்தவுடன் அதற்குரிய கட்டளை வரிகளைச் செய்கிறது. இங்கு ஒவ்வொரு கட்டுப்பாடுகளையும் சரிபார்ப்பதால் , கண்டிப்பாக நேரமானது சற்றே அதிகம் பிடிக்கக் கூடும். இருப்பினும், இதைச் செய்வது கணினி என்பதால் அது பயனருக்குத் தெரிய வாய்ப்பில்லை. ஒரு நிரலரின் **(programmer)** பார்வையில் இதைப்பார்க்கும் பொழுது, கணினியும் விரைவாகச் செயல்பட வேண்டும். விடையும் உகந்த **(optimal answers or optimal results)** முறையில் கிடைக்க வேண்டும். பயனர் தோழமையுடனும் இருக்க வேண்டும் என்பவைகளே முதன்மையாக இருக்க வேண்டும் .

நிரல் **53:**

```
#!/bin/bash
#niral53.sh
#script to find the greatest of the given three numbers
read -p "Enter value of i :" i
read -p "Enter value of j :" j
read -p "Enter value of k :" k
if [ $i -gt $j ]
then
```

```
    if [ $i -gt $k ]
    then
       echo "i is greatest"
    else
       echo "k is greatest"
    fi
else
    if [ $j -gt $k ]
    then
       echo "j is greatest"
    else
        echo "k is greatest"
    fi
fi
```

நிரல் விளக்கம்:

இங்கு இருந்தால் இல்லை கூடு **(nested if statement)** கொண்டு குறுநிரலானது எழுதப்பட்டுள்ளது. இங்கு மூன்று உள்ளீடுகள் பெறப்படுகின்றன. அவை ஒவ்வொன்றாக ஒப்பிட்டுப்பார்க்கப்பட்டு, வெளியீடுகள் தரப்படுகின்றன. இங்கு ஒரு இருந்தால் கட்டளைக்குள்ளே மற்றொரு இருந்தால் கட்டளை கொண்டு அமைவதால், இது கூடு என்று அழைக்கப்படுகிறது. இதுவும், தேர்வுக்கட்டளையை விட மெதுவாகவே செயல்படும்.

நிரல் **54:**

```
#!/bin/bash
#niral54.sh
printf 'Which Linux distribution do you know? '
read DISTR
case $DISTR in
    ubuntu)
        echo "I know it! It is an operating system based on Debian."
        ;;
    centos|rhel)
        echo "Hey! It is my favorite Server OS!"
```

```
        ;;
    windows)
        echo "Very funny..."
        ;;
    *)
        echo "Hmm, seems i've never used it."
        ;;
esac
```

நிரல் விளக்கம்:

இங்கு தேர்வுக்கட்டளையானது மிகவும் எளிமையாகப் பயன்படுத்தப் பட்டிருக்கிறது . இங்கு **$DISTR** என்ற ஒரே ஒரு மதிப்பை மட்டுமே தேர்வுக்கட்டளை கொண்டு செயல்படுகிறது. இங்கு வெவ்வேறான உள்ளீடுகளுக்கு ஏற்றவாறு வெளியீடுகள் கிடைக்கின்றன . ஐந்து முறை நிரலினை இயக்கி அதற்கேற்ற வெளியீடுகள் கீழே கொடுக்கப்பட்டுள்ளன. கண்டிப்பாக இந்த நிரலானது இருந்தால் கட்டளை **(if statement script)** கொண்ட நிரலினை விட விரைவாக இயங்கும். ஏனெனில், இங்கு குறுநிரலின் ஒரு பகுதி இயங்கிய பின் **;;** குறியீடுகள் நிரலினை நிறுத்தி வெளியீட்டினைத் தருகிறது .

நல்ல கைதேர்ந்த நிரலர்கள் **(experienced programmers)** பல இருந்தால் கட்டளைகளைப் **(many if statements)** பயன்படுத்துவதைத் தவிர்த்து , ஒரு தேர்வுக்கட்டளையைப் **(only one case statement)** பயன்படுத்தி நிரல் எழுதி விரைவாக செயல்பாட்டினை முடித்து விடுவார்கள்.

நிரல் வெளியீடு:

```
# ./testcase.sh
Which Linux distribution do you know? centos
Hey! It is my favorite Server OS!
# ./testcase.sh
Which Linux distribution do you know? rhel
Hey! It is my favorite Server OS!
# ./testcase.sh
Which Linux distribution do you know? ubuntu
I know it too! It is an operating system based on Debian.
# ./testcase.sh
```

Which Linux distribution do you know? windows

Very funny...

 # ./testcase.sh

Which Linux distribution do you know? pfff

Hmm, seems i've never used it.

நிரல் **55: (multiple values)**

```
#!/bin/bash
NOW=$(date +"%a")
case $NOW in
    Mon)
        echo "Full backup";;
    Tue|Wed|Thu|Fri)
        echo "Partial backup";;
    Sat|Sun)
        echo "No backup";;
    *) ;;
esac
```

நிரல் விளக்கம்**:**

இங்கு தேர்வுக்கட்டளையிலும் ஒன்றுக்கு மேற்பட்ட மதிப்புக்களைச் சுற்றிவளைத்து கொடுக்கப்பட்டுள்ளது**. |** என்ற குறியீடு அல்லது என்பதைக் குறிப்பதாக அமைகிறது**. $NOW** என்ற மாறியிலும் ஒரு மதிப்பு கணக்கிட்டு இருத்தப்பட்டுள்ளது**.** இதே நிரலினை இருந்தால் கட்டளை **(if statement)** கொண்டு இயக்கிப்பார்க்கவும் **.** சிறிய நிரல் **(small scripts)** என்பதால் நிரலர்களுக்கு நிரல் வெளியீடும் விரைவு தெரியாது**.** இதே கட்டளைகள் பெரிய நிரலுக்குக் **(big scripts)** கொடுக்கப்படும் பொழுது கண்டிப்பாக தேர்வுக் கட்டளையானது **(case statement)** இருந்தால் கட்டளையைக் **(if statement)** காட்டிலும் விரைவாகச் செயல்படுவது தெரியும்**.**

கலைச்சொற்கள்**:**

பற்பல மதிப்புகள் – **multiple values**

இருந்தால் கட்டளை – **if statement**

தேர்வுக் கட்டளை – **case statement**

திண்ணமாய் – sure activity
உகந்த விடைகள் - optimal answers
இருந்தால் இல்லை கூடு - nested if statement
கைதேர்ந்த நிரலர்கள் - experienced programmers
(கற்போம்)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -106- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -15. சுழற்சியில் முறிவுக்கட்டளை (break statement in loop)

நிரலில் செயல்கள் நேர்பட ஓட
வேண்டிய வண்ணம் விடைகள் கிடைத்ததும்
சுழற்சிக் கட்டளை செயலைத் தடுத்து
உந்தி வெளிவர உதவும் முறிவே.
நிரற்பா -15

நிரற்பா விளக்கம்:

குறிப்பிட்ட நிரலில் தேவையான கட்டளைகள் வரிசையில் ஓடி, வேண்டிய விடைகள் கிடைத்ததும், சுழற்சிக் கட்டளையானது **(loop statement – while, until or for loop)** தடுத்து நிறுத்தப்பட்டு அந்தக் குறிப்பிட்ட சுழற்சியிலிருந்து , வெளிவர உதவுவதே முறிவுக் கட்டளையாகும் **(break command).** பின்வரும் நிரல்கள் இவ்வகை முறிவுக்கட்டளைகளை எவ்வாறு பயன்படுத்துவது என்பதை விளக்குவதாக அமைகின்றன.

நிரல் **56:**

```
#!/bin/sh
#niral56.sh
a=0
while [ $a -lt 10 ]
do
  echo $a
  if [ $a -eq 5 ]
  then
    break
  fi
```

```
  a=`expr $a + 1`
done
```

நிரல் விளக்கம்:

இந்த நிரலில் குறிப்பிட்ட சுழற்சிக் கட்டளையானது , **10** முறை செய்ய கட்டளை பிறப்பித்திருந்தாலும் , அது **5** முறை முடிந்ததுமே முறிக்கப்படுகிறது. எனவே வெளியீடானது **5** எண்கள் வரைக்குமே வருகின்றது.

நிரல் வெளியீடு:

```
#./niral56.sh
0
1
2
3
4
5
```

நிரல் **57:**

```
#!/bin/sh
#niral57.sh
for var1 in 1 2 3
do
  for var2 in 0 5
  do
    if [ $var1 -eq 2 -a $var2 -eq 0 ]
    then
      break 2
    else
      echo "$var1 $var2"
    fi
  done
done
```

நிரல் விளக்கம்:

இந்த நிரலிலும் இரண்டு வகையான ஆகக் கட்டளைகள் உள்ளன **(two different for statement).** அதிலிருந்து குறிப்பிட்ட செயல் முடிந்தவுடன் அதே போலவே

[illegible] (break command) [illegible] [illegible] [illegible] [illegible].

[illegible] [illegible]:

```
#./niral57.sh
1 0
1 5
```

[illegible] 58:

```
#!/bin/bash
#niral58.sh
# This script provides wisdom
# You can now exit in a decent way.
FORTUNE=/usr/games/fortune
while true; do
echo "On which topic do you want advice?"
echo "1.  politics"
echo "2.  startrek"
echo "3.  kernelnewbies"
echo "4.  sports"
echo "5.  bofh-excuses"
echo "6.  magic"
echo "7.  love"
echo "8.  literature"
echo "9.  drugs"
echo "10. education"
echo

echo -n "Enter your choice, or 0 for exit: "
read choice
echo
case $choice in
     1)
     $FORTUNE politics
```

```
;;
2)
$FORTUNE startrek
;;
3)
$FORTUNE kernelnewbies
;;
4)
echo "Sports are a waste of time, energy and money."
echo "Go back to your keyboard."
echo -e "\t\t\t\t -- \"Unhealthy is my middle name\" Soggie."
;;
5)
$FORTUNE bofh-excuses
;;
6)
$FORTUNE magic
;;
7)
$FORTUNE love
;;
8)
$FORTUNE literature
;;
9)
$FORTUNE drugs
;;
10)
$FORTUNE education
;;
0)
echo "OK, see you!"
```

```
    break
    ;;
    *)
    echo "That is not a valid choice, try a number from 0 to 10."
    ;;
esac
done
```

நிரல் விளக்கம்:

இங்கு நாம் குறிப்பிட்டுப் பார்க்க வேண்டிய ஒரு அணியானது **(beauty of the break statement),** முறிவுக்கட்டளையானது, ஒரு சுழற்சிக்கட்டளையிலிருந்து மட்டுமே கட்டுப்பாட்டினை வெளிவரச் செய்கிறது ஒரு முழு நிரலிலிருந்து அல்ல என்பதைத் தெளிவு படுத்திக் கொள்ள வேண்டும். இது இந்த நிரலில் நாம் கூடுதலாகக் கொடுத்துள்ள **echo** கட்டளையின் மூலம் செயல் விளக்கம் செய்யப்பட்டுள்ளது. **(This echo will also be executed upon input that causes break to be executed (when the user types "0")).** இந்த நிரலின் வெளியீடானது பயனர் கொடுக்கும் உள்ளீட்டினைப் பொறுத்து, ஒரு சில வரிகள் வரலாம் அல்லது பிழைச் செய்தியாக நாம் **echo** கட்டளையில் கொடுத்துள்ள **(echo "That is not a valid choice, try a number from 0 to 10.")** வெளியீடும் வரலாம்.

நிரல் வெளியீடு:

#./niral58.sh

நிரலினை இயக்கி வேண்டிய வண்ணம் உள்ளீடு கொடுத்து அதற்கேற்றாற் போன்று வரும் வெளியீட்டினைக் காண்க.

கலைச்சொற்கள்:

நேர்பட ஓட – **running sequentially**

சுழற்சிக் கட்டளை – **loop statement**

உந்தி வெளிவர – **jump and come out**

முறிவே – **break**

முறிவுக்கட்டளை – **break statement**

பயனர் உள்ளீடு – **user input**

அணி – **beauty**

கட்டுப்பாடு – **control**

செயல் விளக்கம் – **demonstration**

செயல்பாடு **- execution**

(கற்போம்**)**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -16.

சுழற்சியில் தொடர் கட்டளை அல்லது இடைவிடாக் கட்டளை (continue statement in loop)

பணித்த வேலை பண்புற நிறைவுற
வளைவுக் கட்டளை வழியில் விரைய
இடையில் புகும் இடைவிடாக் கட்டளை
தன்வழி செல்வதை தொடர்ந்து நீட்டுமே.
நிரற்பா -16

நிரற்பா விளக்கம்:

கொடுக்கப்பட்ட வேலையானது நல்ல முறையில் முடிக்கப்பட வளைவுக் கட்டளையானது தனக்கு பணிக்கப்பட்ட வழியில் சென்று கொண்டிருக்கும் பொழுது, இடையில் புகும் தொடர் கட்டளை அல்லது இடைவிடாக் கட்டளை, (continue statement) வளைவுக் கட்டளையினை தன்வழியில் இழுத்து தனக்குப் பிறகு உள்ள கட்டளைகளைச் செய்ய விடாமல், மீண்டும் தொடக்கத்திலிருந்து செய்ய வைக்கும்.

பொதுவமைவு **(General syntax:)**

தொடர் கட்டளையானது தனது பொதுவமைவினை பின்வரும் வண்ணம் இரண்டு வகையாகக் கொண்டுள்ளது.

```
continue
continue n
```

எடுத்துக்காட்டுகள் **(examples):**

```
...
..
for i in something
do
    [ condition ] && continue
    cmd1
    cmd2
```

```
done
..
...
...
..
while true
do
    [ condition1 ] && continue
    cmd1
    cmd2
    [ condition2 ] && break
done
..
...
```

நிரல் **59:**

```
#!/bin/bash
#niral61.sh
x=0
while [ $x -le 5 ]
do
   echo "Before continue : $x"
   x=`expr $x + 1`
   continue
   echo "After continue : $x"
done
echo "While loop finished"
```

நிரல் விளக்கம்**:**

இங்கு தொடர்புக் கட்டளையானது எவ்வாறு ஒரு சிறு நிரலில் பயன்படுத்தப் படுகின்றது என்பது விளக்கப்பட்டுள்ளது**. continue command** வருவதால் **echo "After continue : $x"** இந்தக் கட்டளையின் வெளியீடு வராமலேயே இருக்கிறது**.**

□□□□□□ □□□□□□□□□:

```
Before continue : 0
Before continue : 1
Before continue : 2
Before continue : 3
Before continue : 4
Before continue : 5
While loop finished
```

□□□□□ 60:

```
#!/bin/sh
#niral 59.sh
#mysql backup script
#Must be run as the root user
MUSER="admin"             # MySQL user
MHOST="192.168.1.100"          # MySQL server ip
MPASS="MySQLServerPassword"  # MySQL password

# format dd-mm-yyyy
NOW=$(date +"%d-%m-%Y")

# Backupfile path
BPATH=/backup/mysql/$NOW
# if backup path does not exists, create it
[ ! -d $BPATH ] && mkdir -p $BPATH
# get database name lists
DBS="$(/usr/bin/mysql -u $MUSER -h $MHOST -p$MPASS -Bse 'show
databases')"
for db in $DBS
do
    # Bakcup file name
    FILE="${BPATH}/${db}.gz"
```

```
    # skip database backup if database name is adserverstats or mint
    [ "$db" == "adserverstats"  ] && continue
    [ "$db" == "mint"  ] && continue

    # okay lets dump a database backup
   /usr/bin/mysqldump -u $MUSER -h $MHOST -p$MPASS $db | /bin/gzip
-9 > $FILE
done
```

நிரல் விளக்கம்:

நிரலில் எவ்வாறு தரவுத்தளத்தில் உள்ளவை காப்புப்படி செய்வது என்பது குறித்து கொடுக்கப்பட்டுள்ளது . இங்கு குறிப்பிட்ட இரண்டு பெயர்களில் தரவுத்தளமானது சேமிக்கப்பட்டிருந்தால் அவைகளை விட்டுவிடும்படி பணிக்கப்பட்டுள்ளது. தரவுத்தளங்களின் பெயர்கள் **adserverstats or mint** இருந்தால் அவைகள் காப்புப்படி செய்யப்பட மாட்டாது. ஆனால் இந்த வளைவுக் கட்டளையிலிருந்து கட்டுப்பாடானது வெளிவராமல் தொடர்ந்து அடுத்த தரவுத்தளத்தினை காப்புப்படி எடுக்கச் சென்று விடும் . அதற்காகவே தொடர் கட்டளை இங்கு கொடுக்கப்பட்டுள்ளது.

நிரல் வெளியீடு:

இந்நிரலானது மைஎஸ்கியுஎல் தரவுத்தளத்தினை அடிப்படையாகக் கொண்டு எழுதப்பட்டுள்ளது. ஆதலால், முதலில் தரவுத்தளத்தினை நிறுவுதல் செய்து பின்பு இந்நிரலை இயக்கிப் பார்க்கவும் . இல்லையெனில், பிழைச்செய்தி கிடைக்கும்.

நிரல் **61:**

```
#!/bin/bash
#niral61.sh
# convert all domain names to a lowercase
DOMAINS="$(echo $@|tr '[A-Z]' '[a-z]')"
# Path to named.conf
NAMEDCONF="/var/named/chroot/etc/named.conf"
# Check named.conf for error
NAMEDCHEKCONF="/usr/sbin/named-checkconf -t /var/named/chroot/"
# Display usage and die
```

```
if [ $# -eq 0 ]
then
        echo "Usage: $0 domain1 domain2 ..."
        exit 1
fi

# okay use for loop to process all domain names passed
# as a command line args
for d in $DOMAINS
do
          # if domain  already exits, skip the rest of the loop
        grep $d $NAMEDCONF >/dev/null
        if [ $? -eq 0 ]
        then
                echo "$d exits in in $NAMEDCONF, skiping ..."
                continue # skip it
        fi

        # else add domain to named.conf
        echo "Adding domain $d to $NAMEDCONF..."



        echo "zone \"${d}\" {" >> $NAMEDCONF
        echo "        type master;"  >> $NAMEDCONF
        echo "        file \"/etc/named/master.${d}\";"  >> $NAMEDCONF
        echo "        allow-transfer { slaveservers; };"  >> $NAMEDCONF
        echo "};"  >> $NAMEDCONF

        # Run named configuration file syntax checking tool
        $NAMEDCHEKCONF >/dev/null
        if [ $? -ne 0 ]   # error found?
        then
```

```
        echo "**** Warning: named-checkconf - Cannot reload named due to errors for $d ****"
    else
        echo "**** Domain $d successfully added to $NAMEDCONF ****"

    fi
done
```

நிரல் விளக்கம்:

இங்கு கொடுக்கப்பட்டுள்ள டொமைன்களை (பெயர்களை), பெரிய எழுத்திலிருந்து, சிறிய எழுத்தாக மாற்றும் வண்ணம் நிரல் அமைந்துள்ளது. டொமைனானது, ஏற்கனவே கோப்பில் அமைந்திருப்பின், அதை சுழற்சிக் கட்டளை கண்டுகொள்ளாமல் விட்டுவிட்டு அடுத்த டொமைனைப் பார்க்க வேண்டும். அதற்காக இங்கு தொடர்புக் கட்டளை கொடுக்கப்பட்டுள்ளது. இங்கு **$#** என்பது கட்டளை வரி மதிப்புக்களைக் குறிக்கிறது . நிரலினை இயக்கும் பொழுது பின்வருமாறு கட்டளை கொடுத்து இயக்க வேண்டும்.

#./niral61.sh SOMETHING.COM SOMETHING1.COM

நிரல் வெளியீடு:

டி .என்.எஸ். வழங்கி நிறுவப்பட்டுள்ள பொறிகளில் மட்டுமே இயங்கக் கூடிய ஒரு நிரலாகும். வேறு பொறிகளில் இயக்க பிழைச்செய்தி கிடைக்கும்.

கலைச்சொற்கள்:

பணித்த வேலை – **given task**

பண்புற நிறைவுற – **carried out fully**

வளைவுக் கட்டளை – **loop statement**

இடைவிடாக் கட்டளை – **continue statement**

தொடர்ந்து நீட்டுமே – **to be continued automatically**

காப்புப்படி – **backup**

கட்டளை வரி மதிப்புக்கள் – **command line arguments**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -17 கட்டளைவரி அளவுருக்கள்அல்லதுகட்டளைவரிதருமதிப்புக்கள்

(command line parameters or command line arguments)

அளவுரு கொண்டு அணிசெய முனைவதால் தனிப்பயன் ஆகிய தன்னிரல் வரிகள்
இயக்க நேரம் ஈனும் மதிப்புக்
கொண்டு விடைகள் கொடுத்திட வருமே. நிரற்பா -17

நிரற்பாவிளக்கம் :

அளவுரு கொண்டு ஒரு நிரலை அழகுற அமைக்க முற்படுவதால், பொதுப்பயன் கொண்ட நிரல் வரிகள் தனிப்பயன் கொண்ட வரிகளாக மாறி அனைவருக்கும் பயன்படும் வகையில் அமைவுரும். இயக்க நேரத்தில் கொடுக்கப்படும் மதிப்புக்களை எடுத்துக்கொண்டு, அதற்கேற்றாற்போன்று விடைகள் தரும்.

பொதுவமைவு (General syntax):
கீழே, அளவுருக்கள் பயன்படுத்தும் விதம் அதனது விளக்கங்கள் உள்ளன.
1. $0 இது நிகழும் நிரலினைக் குறிக்கிறது. (The name the script was invoked with. This may be a basename without directory component, or a path name. This variable is not changed with subsequent shift commands.)
2. $1, $2, $3, ... இது ஒன்று இரண்டு மூன்று என்று அளவுருக்களை அல்லது, தருமதிப்புக்களை எடுத்துக் கொள்கிறது. (The first, second, third, ... command line argument, respectively. The argument may contain whitespace if the argument was quoted, i.e. "two words".)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -119- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

3. $# [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] $0 [illegible] [illegible]. (Number of command line arguments, not counting the invocation name $0)

4. $@ [illegible] [illegible] [illegible] [illegible] [illegible]. ("$@" is replaced with all command line arguments, enclosed in quotes, i.e. "one", "two three", "four". Whitespace within an argument is preserved.)

5. $* [illegible] [illegible] [illegible]. [illegible] (including spacebar) [illegible] [illegible] [illegible]. ($* is replaced with all command line arguments. Whitespace is not preserved, i.e. "one", "two three", "four" would be changed to "one", "two", "three", "four". This variable is not used very often, "$@" is the normal case, because it leaves the arguments unchanged.)

[illegible] 62:

```
#!/bin/bash
#niral62.sh
# use predefined variables to access passed arguments
#echo arguments to the shell
echo $1 $2 $3 ' -> echo $1 $2 $3'

# We can also store arguments from bash command line in special
array
args=("$@")
#echo arguments to the shell
echo ${args[0]} ${args[1]} ${args[2]} ' -> args=("$@"); echo $
{args[0]} ${args[1]} ${args[2]}'

#use $@ to print out all arguments at once
echo $@ ' -> echo $@'
```

 # use $# variable to print out
 # number of arguments passed to the bash script
echo Number of arguments passed: $# ' -> echo Number of arguments passed: $#'

நிரல்விளக்கம்:

மேலே உள்ள நிரலில் இயக்க நேரத்தில் தரப்படும் அளவுருக்கள் அல்லது தருமதிப்புக்கள் ஒரு அர்ரே என்றழைக்கக் கூடிய மாறியில் இருத்தி வைக்கப்படுகின்றன.

இங்கு மூன்று தருமதிப்புக்கள் மட்டுமே கொடுக்கப்பட்டுள்ளன. இதே போன்று அதிகமான தருமதிப்புக்களையும் கொடுக்கலாம்.

நிரல்வெளியீடு:

இந்த நிரல் எளிமையானது. எனவே கொடுக்கப்பட்டுள்ள வரிகளைக் கொடுத்து இயக்கிப்பார்த்து விடை காணவும்.

நிரல் **63:**

```
#!/bin/bash
#niral63.sh
echo "Positional Parameters"
echo '$0 = ' $0
echo '$1 = ' $1
echo '$2 = ' $2
echo '$3 = ' $3
```

நிரல்விளக்கம்:

இங்கு மூன்று உள்ளீடுகள் கொடுக்கப்பட்டுள்ளன. அவை பின்வருமாறு

some_program word1 word2 word3
$0 would contain "some_program"
$1 would contain "word1"
$2 would contain "word2"

$3 would contain "word3"

நிரல்வெளியீடு:

நிரலை இயக்கும் பொழுது மூன்று உள்ளீடுகளை பின்வருமாறு கொடுக்க வேண்டும்.

#./niral63.sh one two three

நிரல் **64:**

```
#!/bin/bash
#niral64.sh
if [ "$1" != "" ]; then
   echo "Positional parameter 1 contains something"
else
   echo "Positional parameter 1 is empty"
fi
```

நிரல்விளக்கம்:

இங்கு கொடுக்கப்பட்ட நிரலானது சரியான உள்ளீடுகளை ஏற்றுள்ளதா என இயக்க நேரத்தில் சரிபார்க்கப்படுகிறது .

நிரல்வெளியீடு:

#./niral64.sh

கொடுக்கப்பட்ட உள்ளீட்டினைப் பொறுத்து வெளியீடானது அளிக்கப்படும் .

நிரல் **65:**

```
#!/bin/bash
#niral65.sh
if [ $# -gt 0 ]; then
   echo "Your command line contains $# arguments"
else
   echo "Your command line contains no arguments"
fi
```

நிரல்விளக்கம்:

இந்நிரலானது, அளவுருக்கள் கொடுக்கப்பட்டுள்ளனவா, அவை என்னென்ன என்பதைக் கண்டறியும் வண்ணம் அமைக்கப்பட்டுள்ளது .

[illegible] 65:

[illegible].

```
#!/bin/bash
#niral66.sh
echo "You start with $# positional parameters"

# Loop until all parameters are used up
while [ "$1" != "" ]; do
    echo "Parameter 1 equals $1"
    echo "You now have $# positional parameters"

    # Shift all the parameters down by one
    shift
done
```

[illegible]:

[illegible] - **command line parameters**

[illegible]- **command line arguments**

[illegible] – **parameters**

[illegible] – **decorate or write**

[illegible] – **customized**

[illegible] – **personal script**

[illegible] – **run time**

[illegible] – **given value**

[illegible] - **output**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 18.கட்டளைவரி அளவுருக்கள் அல்லது கட்டளைவரி தருமதிப்புக்கள் (command line parameters or command line arguments) - தொடர்ச்சி

தனியர் செய்த தவறிலா நிரல்கள்
நல்ல பயனை நீட்டித் தந்திடின்
பயன்பாடு அனைத்தும் பயனர் பெற்றிட
அளவுருக் கொண்டு அமைத்திட வருமே.
- நிரற்பா **18**

நிரற்பாவிளக்கம்:

தனியாக ஒருவர் செய்யும் பிழையற்ற நிரலானது, நாளடைவில் மாறாமல் நீடித்து உழைக்குமெனில் அந்த நிரல் கொடுக்கும் பயன்பாட்டினை அனைத்துப் பயனர்களும் பெற்றிட அளவுரு கொண்டு அமைத்திடல் வேண்டும்.

நிரல் **66:**

கீழ்க்காணும் நிரலானது மிகவும் பெரிய நிரலாகும் . இந்த நிரலானது, ஒரு பொறியின் தகவல்களைத்திரட்டித் தருவதாக அமைகிறது. இந்தத் தகவல்கள் அனைத்தும் ஒரு **HTML** கோப்பாக அமைகின்றன.

இது பெரிய நிரலாகையால், அவை பகுக்கப்பட்டு சிறுசிறு நிரல் துண்டுகளாகக் **(functions)** கொடுக்கப்பட்டுள்ளது. ஒவ்வொரு நிரல் துண்டும் ஒவ்வொரு வேலையைச் செய்கிறது. குறிப்பிட்ட வேலையைச் செய்து முடித்தவுடன் **main** க்கு கட்டுப்பாட்டினைக் கொடுக்கிறது.

#!/bin/bash

```
#niral66.sh
# system_page - A script to produce a system information HTML file
##### Constants

TITLE="System Information for $HOSTNAME"
RIGHT_NOW=$(date +"%x %r %Z")
TIME_STAMP="Updated on $RIGHT_NOW by $USER"



##### Functions

function system_info
{
    echo "<h2>System release info</h2>"
    echo "<p>Function not yet implemented</p>"

}   # end of system_info
```

Command Line Arguments

- Shell scripting can accept command line arguments.
- Within a shell script, you can refer to these args as $1, $2, $3 and so
- $# - Number of command line arguments
- $* - Display all the arguments
- $0 – Name of the script

 58

```
function show_uptime
{
   echo "<h2>System uptime</h2>"
   echo "<pre>"
   uptime
   echo "</pre>"

}   # end of show_uptime



function drive_space
{
   echo "<h2>Filesystem space</h2>"
   echo "<pre>"
```

```
    df
    echo "</pre>"

}   # end of drive_space



function home_space
{
    # Only the superuser can get this information

    if [ "$(id -u)" = "0" ]; then
        echo "<h2>Home directory space by user</h2>"
        echo "<pre>"
        echo "Bytes Directory"
        du -s /home/* | sort -nr
        echo "</pre>"
    fi

}   # end of home_space



function write_page
{
    cat <<- _EOF_
    <html>
        <head>
        <title>$TITLE</title>
        </head>
        <body>
        <h1>$TITLE</h1>
        <p>$TIME_STAMP</p>
        $(system_info)
```

```
        $(show_uptime)
        $(drive_space)
        $(home_space)
        </body>
    </html>
_EOF_

}



function usage
{
    echo "usage: system_page [[[-f file ] [-i]] | [-h]]"
}




##### Main

interactive=
filename=~/system_page.html

while [ "$1" != "" ]; do
    case $1 in
        -f | --file )           shift
                                filename=$1
                                ;;
        -i | --interactive )    interactive=1
                                ;;
        -h | --help )           usage
                                exit
                                ;;
```

```
        * )                 usage
                            exit 1
   esac
   shift
done




# Test code to verify command line processing

if [ "$interactive" = "1" ]; then
     echo "interactive is on"
else
     echo "interactive is off"
fi
echo "output file = $filename"


# Write page (comment out until testing is complete)

# write_page > $filename
```

இது ஒரு சிறு திட்ட அறிக்கை **(mini-project)** போன்றது**.** இங்கு பல நிரல்கள் ஒன்றுடன் ஒன்று பிணைந்து ஒரு பெரிய நிரலாகக் காணப்படுகிறது**.** இங்கு பொதுவாக நாம் லினக்சு இயங்கு தளத்தில் பயன்படுத்தும் கட்டளைகளைப் பயன்படுத்தி**,** அதன் மூலமாக வரும் வெளியீட்டினை **HTML** கோப்பாக மாற்றி வெளியீடானது தரப்படுகிறது**.** பொதுவாக பெரிய திட்ட அறிக்கைகள் **(major projects)** கூட இம்முறையிலேயே தயாரிக்கப்படுகின்றன**.**

நாம் பயன்படுத்தும் அனைத்து வகையான குறு நிரல்களையும்**,** இதுபோன்று அளவுருக்கள் **(parameters)** அல்லது தருமதிப்புக்கள் **(arguments)** கொண்டு அமைத்திட்டால் எல்லாப் பயனரும் பயன்படுத்தி விடையறிய இயலும்**.**

கலைச்சொற்கள் :

தனியர் – single user
நல்ல பயனை நீட்டித் தந்திடின் – working for a long run பயன்பாடு – usage, utilization
பயனர் – user
அளவுரு – parameters
சிறு திட்ட அறிக்கை – mini project
பெரிய திட்ட அறிக்கைகள் – major projects
(கற்போம்)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -131- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -19. உறக்கக் கட்டளை அல்லது தூக்கக் கட்டளை (sleep command)

ஒன்றன் பின்னே ஒன்றாய் வந்திடும் வரிகள் தாமே வரிசை நிற்க
இடைவெளி விட்டு எழுந்து தூங்கி
வேலை செய்ய வைத்திடும் உறக்கமே.
-நிரற்பா 19

நிரற்பாவிளக்கம் :

ஒன்றன் பின் ஒன்றாக வரிசையான முறையில் அமைந்த கட்டளை வரிகள், குறித்த கால இடைவெளியினை விட்டு மீண்டும் விட்ட வரிகளைத் தொடர்ந்து செய்ய உதவி செய்வதே தூக்கக் (sleep command) கட்டளையாகும்.
General syntax: (பொது அமைவு)
தூக்கக் கட்டளையின் பொது அமைவினை காணலாம்.
sleep NUMBER[SUFFIX]
Where SUFFIX may be:
s for seconds (the default)
m for minutes.
h for hours.
d for days.

sleep --help
sleep --version
இந்தக் கட்டளையின் வாயிலாக, நாம் ஒரு நொடியிலிருந்து சில நாட்கள் வரை தூங்க வைக்க (sleep state) முடியும்.
To sleep for 5 seconds, use:
sleep 5

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -132- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

To sleep for 2 minutes, use:

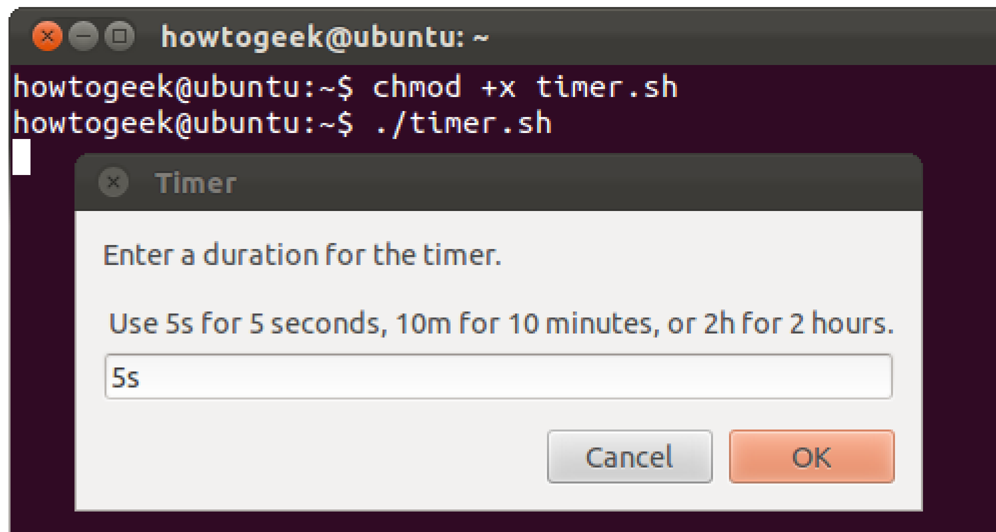
sleep 2m

To sleep for 3 hours, use:

sleep 3h

To sleep for 5 days, use:

sleep 5d



[illegible] 67:

```
#!/bin/bash
#niral67.sh
#this script explains how sleep works in shell script
echo "Hi, I'm sleeping for 5 seconds..."
sleep 5
echo "all Done."
```

[illegible]:

[illegible]. [illegible], [illegible] all Done [illegible].

./niral67.sh

[illegible] 68:

```
#!/bin/bash
#niral68.sh
## run commmand1, sleep for 1 minute and finally run command2 ##
command1 && sleep 1m && command2
```

```
## sleep in bash for loop ##
for i in {1..10}
do
  do_something_here
  sleep 5s
done
```

நிரல் விளக்கம்:

இந்த நிரலிலே, **sleep** கட்டளையானது எவ்வாறு **for** என்னும் வளைவுக் கட்டளையிலே பயன்படுத்தப்படுகிறது என்பது விளக்கப்பட்டுள்ளது. **do_something_here** என்ற இடத்திலே நமக்குத் தேவையான வரிகளை உள்ளிட்டு நிரலினை நிரப்பலாம்.

நிரல் **69:**

```
#!/bin/bash
#niral69.sh
# this script uses sleep command
## run while loop to display date and hostname on screen ##
while [ : ]
do
   clear
   tput cup 5 5
   date
   tput cup 6 5
   echo "Hostname : $(hostname)"
   sleep 1
done
```

நிரல் விளக்கம்:

./niral68.sh என்ற கட்டளை மூலம் இயக்கிப்பார்ப்பின், நிரலானது திரையில் நேரம் **(time)** மற்றும் பொறியின் பெயர் **(hostname)** ஆகியவற்றை ஒரு நொடி கால இடைவெளியில் மேம்படுத்துவதைக்**(update)** காணலாம். இங்கு **tput cup 5 5** என்பது திரையில் வெளியீடு வெளியிடப்பட வேண்டிய இடத்தினைக் குறிக்கிறது . **sleep 1** என்பது ஒரு நொடித் தூக்கத்தைக் குறிக்கிறது .

நிரல் **69:**

```
#!/bin/bash
#niral69.sh
#run another script in the main script
i=1
while [ "$i" -ne 0 ]
do
  i=./runEmailAgent
  sleep 10
done
```

நிரல் விளக்கம்:

இங்கு நாம் ஏற்கனவே எழுதிய வேறொரு குறுநிரலை **(script),** குறிப்பிட்ட கால இடைவெளியில் திரும்பத்திரும்ப அழைக்கும் முறையானது விளக்கப்பட்டுள்ளது. இதில், **./runEmailAgent** என்பது ஏற்கனவே எழுதியமைக்கப்பட்ட வேறொரு நிரலாகும். அதை நாம் இந்நிரலில் ஒரு வளைவுக்கட்டளையினுள் அழைக்கிறோம்.

நிரல் **70:**

```
#!/bin/sh
#niral76.sh
#shell script example
before="$(date +%s)"
echo $before
sleep 3
after="$(date +%s)"
echo $after
elapsed_seconds="$(expr $after - $before)"
echo Elapsed time for code block: $elapsed_seconds
```

நிரல் விளக்கம்:

இதுவும் **sleep** கட்டளையினைப் பயன்படுத்திச் செய்யப்படும் ஓர் எடுத்துக்காட்டு நிரலாகும் . இதனைச் செய்து பார்த்து விடையறிய முயலுக.

./niral70.sh

குறிப்பு : இந்த நிரல்கள் அனைத்தும் அனைத்து வகையான இலினக்சு இயங்குதளங்களிலும் **(Red Hat, Ubuntu, Suse Linux, Debian, Pinguy, Linux**

mint) இயங்கும்.

கலைச்சொற்கள் :

தனியர் – single user
நல்ல பயனை நீட்டித் தந்திடின் – working for a long run
பயன்பாடு – usage, utilization
பயனர் – user
அளவுரு – parameters
சிறு திட்ட அறிக்கை – mini project
பெரிய திட்ட அறிக்கைகள் – major projects
(கற்போம்)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -136- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -20. அளவுருக்கள் அல்லது தருமதிப்புக்கள் (command line arguments or command line parameters) தொடர்ச்சி

ஒன்றாய்ச் சுழியம் ஒன்பது இறுவாய்
எண்ணைச் சுருளாய் எதையும் அழைத்திடும்
எண்ணுதல் நிகழெண் ஒழுங்கு வெளிநிலை
பின்னெண் என்றெலாம் பூத்திடும் அளவுரே!
- நிரற்பா 20

நிரற்பாவிளக்கம்:

சுழியம் தொடங்கி ஒன்பது முடிய அனைத்து வகையான எண்களும் **$** என்ற குறியீட்டுடன் பயன்படுத்தி அவற்றின் மதிப்புக்களை நிரலில் இருத்தலாம். ஒன்பதிற்கு மேல் உள்ள எண்களை அழைக்கும் பொழுது சுருள் அடைப்புக்குறிகளுக்குள் அழைக்க வேண்டும். சுருள் அடைப்புக்குறிகளுக்குள் **(curly braces) { }** அழைக்க வேண்டும். அளவுருக்களின் எண்ணிக்கை **$#** என்ற குறிப்பிலும் அளவுரு ஒழுங்கு **$@** என்பதிலும், நிகழ் செயல்முறை எண் அல்லது நிகழெண் **$$** என்ற குறிப்பிலும் , வெளியேறும் நிலை **$?** என்பதிலும், கடந்த பின்புல செயல்முறை எண் (பின்னெண்) **$!** என்பதிலும் உள்ளிருப்பாக இருத்தப்பட்டிருக்கும் .

நிரல் **71**

```
#!/bin/bash
#niral71.sh
If [ $# = 3 ]
Then
echo “$@”
else
```

echo “Args are not three”

fi

நிரல் விளக்கம்:

இந்த நிரலில் , கொடுக்கப்பட்ட அளவுருக்கள் எந்த வகையில் வந்துள்ளன. அளவுருக்களின் எண்ணிக்கை மூன்று இருக்கிறதா என்பது அறியப்படுகிறது.

நிரல் **72**

```
#!/bin/bash
#niral72.sh
echo “Provide user name”
read i
a = `w | awk ‘{print $1}’ | sort –n | uniq | grep $i`
if [ “$i” = “$a” ]
then
echo “user logged in”
else
echo “user not logged in”
fi
```

நிரல் விளக்கம்:

கொடுக்கப்பட்ட நிரலில், நாம் தரும் பயனர் பெயரானது, இப்பொழுது உள்நுழைந்து இருக்கிறதா என்பது அறியப்படுகிறது. இங்கு நாம் குறிப்பிட்டு அறிய வேண்டிய கட்டளை வரி **a = `w | awk ‘{print $1}’ | sort –n | uniq | grep $i`**

இதுதான்.

நிரல் **73 -** நிரல் **72 (**பதிப்பு **2)**

```
#!/bin/bash
#niral73.sh
echo –n “Please enter a username:”
read user
a=`grep $user /etc/passwd`
b=`w | awk ‘{print $1}’ | grep $user`
if [ ! “$a” ]
then
```

```
echo "$user is not a valid user"
exit
elif [ "$b" ]
then
echo "$user is logged in"
else
echo "$user is not logged in"
fi
```

நிரல் விளக்கம்:

இது **72** ஆம் நிரலில் வரும் அதே வேலையைத்தான் செய்கிறது. ஆனால் இங்கு அளவுருவானது **(arguments or parameters)** கையாளப்படாமல் மாறியானது **(variable)** கையாளப்படுகிறது.

நிரல் **74**

```
#!/bin/bash
#niral74.sh
#script used to file exists or not
a=$1
if [ -e "$a" ]
then
echo "$a file exist & modification time is" `ls –ltr $a | awk '{print $8}'`
else
echo "$a file doesn't exist"
fi
```

நிரல் விளக்கம்:

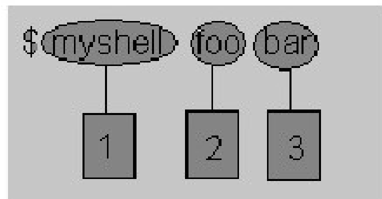
இந்நிரலில், அளவுரு அல்லது தருமதிப்பின் மூலம் கொடுக்கப்படும் கோப்பானது குறிப்பிட்ட இடத்தில் இருக்கிறதா என கண்டறியப்படுகிறது.

Why Command Line arguments required

1. Telling the command/utility which option to use.
2. Informing the utility/command which file or group of files to process (reading/writing of files).

Let's take rm command, which is used to remove file, but which file you want to remove and how you will tell this to rm command (even rm command don't ask you name of file that you would like to remove). So what we do is we write command as follows:
$ rm {file-name}
Here rm is command and filename is file which you would like to remove. This way you tail rm command which file you would like to remove. So we are doing one way communication with our command by specifying filename. Also you can pass command line arguments to your script to make it more users friendly. But how we access command line argument in our script.

Lets take ls command
$ Ls -a /*
This command has 2 command line argument -a and /* is another. For shell script,
$ myshell foo bar



1 Shell Script name i.e. myshell

2 First command line argument passed to myshell i.e. foo

3 Second command line argument passed to myshell i.e. bar

In shell if we wish to refer this command line argument we refer above as follows

1 myshell it is $0

2 foo it is $1

2 bar it is $2

□□□□□ 75

#!/bin/bash

#niral75.sh

 # Call this script with at least 10 parameters, for example

 # ./scriptname 1 2 3 4 5 6 7 8 9 10

MINPARAMS=10

```
echo

echo "The name of this script is \"$0\"."
# Adds ./ for current directory
echo "The name of this script is \"`basename $0`\"."
# Strips out path name info (see 'basename')

echo

if [ -n "$1" ]              # Tested variable is quoted.
then
 echo "Parameter #1 is $1"  # Need quotes to escape #
fi

if [ -n "$2" ]
then
 echo "Parameter #2 is $2"
fi

if [ -n "$3" ]
then
 echo "Parameter #3 is $3"
fi
# ...you can add your own comments here for further reference
if [ -n "${10}" ]  # Parameters > $9 must be enclosed in {brackets}.
then
 echo "Parameter #10 is ${10}"
fi

echo "-----------------------------------"
echo "All the command-line parameters are: "$*""
```

```
if [ $# -lt "$MINPARAMS" ]
then
  echo
  echo "This script needs at least $MINPARAMS command-line arguments!"
fi

echo
exit 0
```

நிரல் விளக்கம்:

குறிப்பிட்ட நிரலில் குறைந்த அளவு பத்து அளவுருக்களாவது பயன்படுத்த வேண்டும். **(# Call this script with at least 10 parameters, for example)** இல்லையெனில் நிரலில் எதிர்பார்த்த வெளியீடு **(output)** கிடைக்காது.

கலைச்சொற்கள்**:**

சுழியம் – **Zero**

ஒன்றாய் – முதலாய் **(first)**

இறுவாய் – **last**

எண்ணைச் சுருளாய் – **number in curly braces eg {333}**

எதையும் அழைத்திடும் – **call anything**

எண்ணுதல் **(**எண்ணிக்கை**) – count ($#)**

நிகழெண் **(**நிகழ் செயல்முறை எண் **$$) – current process id**

ஒழுங்கு **- $@ arguments in the given order**

வெளிநிலை – **exit status**

பின்னெண் **-** கடந்த பின்புல செயல்முறை எண் **$! – Previous process id**

அளவுரே **-** அளவுருவே

(கற்போம்**)**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் -21

தக்க நொடியில் தீர்வுற அமைந்திடும்
பொதுவாய்ச் செய்திடும் பற்பல செயல்கள்
வரிகள் எங்ஙனம் வந்தன என்பதைப்
பொறுத்தே அமைந்திடல் பயக்கும் நலமே.
-நிறற்பா 21

நிரற்பாவிளக்கம்:

தேவையானவற்றை சரியான நேரத்தில் செய்து, விடைகளை அளித்து சரியான முறையில் செயல்களை நடத்திட உதவுதல், ஒரு நிரலில் உள்ள கட்டளைவரிகள் எவ்வாறு அமைகின்றன என்பதைப் பொறுத்தே அமைகிறது.

நிரல் **76**

```
#!/bin/bash
#niral76.sh
mkdir .recyclebin
mv $@ .recyclebin
echo –n “Please enter the filename to delete:”
read file
mv $file .recyclebin
```

நிரல் விளக்கம்:

இந்நிரலானது ஒரு கட்டளைவரி ரீசைக்கிள் பின் **(recycle bin)** போன்று செயல்படுகிறது. எந்த ஒரு கோப்பினையும் அழிப்பதற்கு நாம் **rm** கட்டளையினைப் பயன்படுத்துவோம். இங்கு **.recyclebin** என்பது ஒரு பொதுவான அடைவு ஆகும். ஆனால் அது ஒரு **recyclebin** போன்று செயல்படுகிறது.

நிரல் **77**

```
#!/bin/bash
#niral77.sh
a=$1
if [ “$a” = L ]
```

```
then
ls –l .recyclebin
fi
if [ “$a” = D ]
then
echo –n “Please enter the filename to delete:”
read file
mv $file .recyclebin
fi
```

நிரல் விளக்கம்:

இங்கும் மேற்கூறிய நிகழ்வானது வேறு ஒரு வண்ணம் செயல்படுகிறது. இந்நிரல் **recycle bin** நிரலின் மற்றொரு பதிப்பாக எடுத்துக் கொள்ளலாம்.

நிரல் **78**

```
#!/bin/bash
#niral78.sh
#this program used to convert the image format into png

for i in *.pcx ; do
   CMD="convert -quality 625 $i `echo $i | sed -e 's/\.pcx$/.png/'`"
# Show the command-line to the user:
   echo $CMD
# Execute the command-line:
   eval $CMD
done
```

நிரல் விளக்கம்:

இந்த நிரலானது, **ImageMagick** என்னும் பொதியினை உள்ளெடுத்து அமைந்துள்ளது. இப்பொதியானது, கொடுக்கப்பட்ட படத்தினை, வேறொரு கோப்பு அமைவில் **(changing to png format)** மாற்றித்தர உதவுகிறது. இது ஒரே கட்டளைவரி மூலம் பல படங்களின் கோப்பு அமைவுகளை மாற்றித் தருகிறது .

நிரல் **79**

```
#!/bin/bash
#niral79.sh
```

#To erase a file proper, requires writing random bytes into the disk blocks occupied by the file.

```
for i in * ; do
   dd if=/dev/urandom    \
      of="$i"            \
      bs=1024            \
      count=`expr 1 +    \
        \`stat "$i" | grep 'Size:' | awk '{print $2}'\`  \
          / 1024`
done
```

நிரல் விளக்கம்**:**

ஒரேஒரு கட்டளைவரியானது புரிந்து கொள்வதற்கு ஏதுவாக, மடித்து மடித்துக் கொடுக்கப்பட்டுள்ளது. இங்கே ஒரு குறிப்பிட்ட கோப்பில் **Random** எண்ணானது இருத்தப்பட்டு ஏற்கனவே இருக்கும் செய்திகள் அழிக்கப்படுகின்றன. இது **disk blocks** களை அழிக்கப்பயன்படுகிறது.

நிரல் **80**

```
#!/bin/sh
#niral80.sh
# Get the files:
FILES=`ls -1`

for FILE in $FILES
do
    IDX=`expr index $FILE .`

    if [ "$IDX" == 0 ]; then
        IDX=`expr length $FILE`
    else
        IDX=`expr $IDX - 1`
    fi

    SUB=`expr substr $FILE 1 $IDX`
```

```
    echo "Sub File: $SUB"
done
```

நிரல் விளக்கம்:

நிரலானது, குறிப்பிட்ட **string** ஐ ஒரு கோப்பில் தேட உதவுகிறது. நிரலினைத் தட்டச்சுச் செய்து வெளியீடு அறிய முயலுக.

நிரல் 81

கீழ்க்காணும் இந்நிரலைச் செய்து பார்த்து விடையறிய முயலுக. இந்நிரலானது, வருகைப்பதிவுகள் எனப்படும் **log** கோப்புக்களைக் கண்டு அவற்றை முழுவதுமாக அழிக்கிறது. பொதுவாகக் கோப்புக்கள் அழிக்கப்படும் பொழுது அவற்றின் உத்தரவு மட்டுமே பயனருக்கு மறுக்கப்படுகின்றது. முழுமையாக அவை அழிக்கப்படுவதில்லை. எனவே வேறு ஏதேனும் மென்பொருட்கள் மூலம் இழந்த தரவுகள் மீட்கப்படுகின்றன. இந்நிரல் மூலம் நாம் முழுமையாகக் கோப்புக்களை அழிக்க முடியும் .

```
#!/bin/bash
# niral81.sh
#  Warning:
#  This script uses quite a number of features that will be explained
#+ later on.
#  By the time you've finished the first half of the book,
#+ there should be nothing mysterious about it.
LOG_DIR=/var/log
ROOT_UID=0     # Only users with $UID 0 have root privileges.
LINES=50       # Default number of lines saved.
E_XCD=86       # Can't change directory?
E_NOTROOT=87   # Non-root exit error.
# Run as root, of course.
if [ "$UID" -ne "$ROOT_UID" ]
then
  echo "Must be root to run this script."
  exit $E_NOTROOT
fi
```

```
if [ -n "$1" ]
# Test whether command-line argument is present (non-empty).
then
  lines=$1
else
  lines=$LINES # Default, if not specified on command-line.
fi
#  Stephane Chazelas suggests the following,
#+ as a better way of checking command-line arguments,
#+ but this is still a bit advanced for this stage of the tutorial.
#
#    E_WRONGARGS=85  # Non-numerical argument (bad argument
format).
#
#    case "$1" in
#    ""      ) lines=50;;
#    *[!0-9]*) echo "Usage: `basename $0` lines-to-cleanup";
#     exit $E_WRONGARGS;;
#    *       ) lines=$1;;
#    esac
#
#* Skip ahead to "Loops" chapter to decipher all this.
cd $LOG_DIR
if [ `pwd` != "$LOG_DIR" ]  # or   if [ "$PWD" != "$LOG_DIR" ]
                            # Not in /var/log?
then
  echo "Can't change to $LOG_DIR."
  exit $E_XCD
fi  # Doublecheck if in right directory before messing with log file.

# Far more efficient is:
#
```

```
# cd /var/log || {
#   echo "Cannot change to necessary directory." >&2
#   exit $E_XCD;
# }
tail -n $lines messages > mesg.temp # Save last section of message
log file.
mv mesg.temp messages               # Rename it as system log file.
#  cat /dev/null > messages
#* No longer needed, as the above method is safer.
cat /dev/null > wtmp  #  ': > wtmp' and '> wtmp'  have the same
effect.
echo "Log files cleaned up."
#  Note that there are other log files in /var/log not affected
#+ by this script.

exit 0
#  A zero return value from the script upon exit indicates success
#+ to the shell.
```

[illegible]:

[illegible] – **proper time**

[illegible] – **solving issues**

[illegible] – **common tasks**

[illegible] – **order of the commands**

[illegible] - **package**

([illegible])

– 22
(shift command)

\- 22

:

Shell Scripting: Program Arguments : shift command

Shift

The shift command can be used to shift arguments to left side.

We can specify a count and we lose that many arguments on the left side. For a shift of 1, $2 becomes $1 and so on.

For a shift of 2, $3 will become $1.

It is useful to process arguments in a loop using a single variable to reference to argument one by one.

www.opengurukul.com 35

நிரல் **82**

```
#!/bin/bash
#shift command in positional parameters
#niral82.sh
echo "Current command line args are: \$1=$1, \$2=$2, \$3=$3"
shift
echo "After shift command the args are: \$1=$1, \$2=$2, \$3=$3
```

நிரல் விளக்கம்**:**

இது மாற்றக் கட்டளையினை எளிதில் புரிந்து கொள்வதற்காக அமைக்கப்பட்டுள்ள எளிமையான நிரலாகும்**.** முதலில் அளவுருக்கள் கொடுக்கப்பட்ட வரிசையிலேயே வருகின்றன**.** மாற்றக் கட்டளையினை கொடுத்த பிறகு**,** முதல் அளவுரு விடப்பட்டு**,** இரண்டாம் அளவுருவிலிருந்து எடுத்துக் கொள்ளப்படுகிறது**.**

Excute above script as follows:

```
# chmod +x shiftdemo.sh
# ./shiftdemo -f foo bar
```

Current command line args are: $1=-f, $2=foo, $3=bar

After shift command the args are: $1=foo, $2=bar, $3=

நிரல் **83**

```
#!/bin/bash
#niral83.sh
#using shift command
while [ "$1" ]
do
   if [ "$1" = "-b" ]; then
        ob="$2"
        case $ob in
          16) basesystem="Hex";;
           8) basesystem="Oct";;
           2) basesystem="bin";;
           *) basesystem="Unknown";;
        esac
       shift 2
```

```
  elif [ "$1" = "-n" ]
  then
    num="$2"
    shift 2
  else
    echo "Program $0 does not recognize option $1"
    exit 1
  fi
done
output=`echo "obase=$ob;ibase=10; $num;" | bc`
echo "$num Decimal number = $output in $basesystem number system(base=$ob)""
```

[illegible]:

[illegible] shift 2 [illegible], [illegible]. [illegible].

```
# chmod +x convert
# ./convert -b 16 -n 500
500 Decimal number = 1F4 in Hex number system(base=16)
# ./convert -b 8 -n 500
500 Decimal number = 764 in Oct number system(base=8)
# ./convert -b 2 -n 500
500 Decimal number = 111110100 in bin number system(base=2)
# ./convert -b 2 -v 500
Program ./convert does not recognize option -v
# ./convert -t 2 -v 500
Program ./convert does not recognize option -t
# ./convert -b 4 -n 500
500 Decimal number = 13310 in Unknown number system(base=4)
# ./convert -n 500 -b 16
```

500 Decimal number = 1F4 in Hex number system(base=16)

[illegible] 84:

```
#!/bin/bash
#niral84.sh
# This script can clean up files that were last accessed over 365 days
ago.

USAGE="Usage: $0 dir1 dir2 dir3 ... dirN"

if [ "$#" == "0" ]; then
    echo "$USAGE"
    exit 1
fi

while (( "$#" )); do

if [[ $(ls "$1") == "" ]]; then
    echo "Empty directory, nothing to be done."
  else
    find "$1" -type f -a -atime +365 -exec rm -i {} \;
fi

shift

done
```

[illegible] [illegible]:

[illegible] [illegible], [illegible] 365 [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible], [illegible]

கொடுக்கப்பட்ட அடுத்தடுத்த அடைவுகளை எடுத்தாளப்பயன்படுகிறது. இத்தகைய நிரல்கள் தானியங்கு நிரல்களாக நிறையப் பயன்படுத்தப்படுகின்றன.

நிரல் **85:**

```
#!/bin/bash
#niral85.sh
if [ $# -lt 1 ]; then
        echo "Usage: $0 package(s)"
        exit 1
fi
while (($#)); do
    yum install "$1" << CONFIRM
y
CONFIRM
shift
done
```

நிரல் விளக்கம்:

இந்நிரல் மூலமாக நாம் ஒன்றிற்கு மேற்பட்ட பொதிகளை விரைவாக நிறுவ இயலும். **(This is used to install multiple packages at once.)** ஒவ்வொரு பொதிகளாக நிறுவிய பின்னர் தருமதிப்பாகக் கொடுக்கப்பட்ட அடுத்த பொதியினை மாற்றக் கட்டளை மூலமாக நிரல் எளிதில் அறிந்து கொள்ளும் வண்ணம் நிரல் அமைக்கப்பட்டுள்ளது.

கலைச்சொற்கள்:

அமைந்த வண்ணம் – **in the given order**

அளவுரு இருக்க – **available parameter**

நிரலின் பயன் – **usage of the script**

நெடிதாய் ஆகுமாம் – **expanding the usage**

உற்றுழி – **on emergency**

மாற்றே – **shift command**

அடைவுகள் **- folders**

(கற்போம்**)**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 23 Trap command (கண்ணி -கட்டளை)

தங்கு தடையறத் தீர்வு தந்திடும்
நிரலினை குறித்த நிகழ்செயல் செய்து
முடிந்ததும் இடையினில் மறித்து நிகழும்
தாடை நீக்கித் தவிர்க்கும் கண்ணியே.
- நிரற்பா 23

நிரற்பாவிளக்கம்:

முறையாகச் செயல்பட்டுக் கொண்டிருக்கும் நிரலினை, ஒரு குறிப்பிட்ட செயற்பாடு முடிந்தவுடன் ஏதேனும் ஒரு தாடையைப் அதாவது **signal** ஐ கண்ணி **(trap command)** கட்டளையினைப் பயன்படுத்தி நிறுத்த முடியும்.

Syntax (பொது அமைவு**)**

இக்கட்டளையின் பொது அமைவு பின்வருமாறு அமைகிறது.

```
trap arg signal
trap command signal
trap 'action' signal1 signal2 signalN
trap 'action' SIGINT
trap 'action' SIGTERM SIGINT SIGFPE SIGSTP
trap 'action' 15 2 8 20
```

நிரல் **86**

```
#!/bin/bash
#niral86.sh
# capture an interrupt # 0
trap 'echo "Exit 0 signal detected..."' 0

# display something
echo "This is a test"
```

exit shell script with 0 signal
exit 0

நிரல் விளக்கம்:

இது ஓர் எடுத்துக்காட்டான நிரல் . இது எவ்வாறு ஒரு தாடையினைக் கண்ணி வைத்துப் பிடிப்பது என்பது பற்றி அறிய உதவுகிறது. கீழ்க்காணுமாறு இதனது **permission** மாற்றி இதனை எளிமையாக இயக்கி விடையறியலாம்.

chmod +x testtrap.sh

./testtrap.sh

வெளியீடு:

This is a test

Exit 0 signal detected....

நிரல் **87**

```
#!/bin/bash
#niral87.sh
# Capture an interrupt # 2 (SIGINT)
trap '' 2
# read CTRL+C from keyboard with 30 second timeout
read -t 30 -p  "I'm sleeping hit CTRL+C to exit..."
```

நிரல் விளக்கம்:

இதுவும் கூட, மேற்கூறிய நிரல் போன்றே செயல்படுகிறது. **CTRL+C** கொடுக்காவிடில், தொடர்ந்து முப்பது நொடிகளுக்கு வேறு எதுவும் உள்ளிட முடியாது.

I'm sleeping hit CTRL+C to exit...^C^C^C^C

Command

- trap action after receiving signal

  `trap command signal`

- signal explain

signal	explain
HUP (1)	hung up
INT (2)	interrupt (Crtl + C)
QUIT (3)	Quit (Crtl + \)
ABRT (6)	Abort
ALRM (14)	Alarm
TERM (15)	Terminate

42

□□□□□ 88:

```
#!/bin/bash
#niral88.sh
# Program to print a text file with headers and footers

TEMP_FILE=/tmp/printfile.txt

function clean_up {

    # Perform program exit housekeeping
    rm $TEMP_FILE
    exit
}

trap clean_up SIGHUP SIGINT SIGTERM
```

```
lpr $1 > $TEMP_FILE
echo -n "Print file? [y/n]: "
read
if [ "$REPLY" = "y" ]; then
    lpr $TEMP_FILE
fi
clean_up
```

[illegible]:

[illegible], [illegible] header [illegible] footer [illegible]. clean_up [illegible] (function), [illegible].

[illegible].

[illegible] 89:

```
#!/bin/bash
#niral89.sh
#another version of the previous script
# Program to print a text file with headers and footers

# Usage: print file

# Create a temporary file name that gives preference
# to the user's local tmp directory and has a name
# that is resistant to "temp race attacks"

if [ -d "~/tmp" ]; then
    TEMP_DIR=~/tmp
else
    TEMP_DIR=/tmp
fi
TEMP_FILE=$TEMP_DIR/printfile.$$.$RANDOM
PROGNAME=$(basename $0)
```

```
function usage {

    # Display usage message on standard error
    echo "Usage: $PROGNAME file" 1>&2
}

function clean_up {

    # Perform program exit housekeeping
    # Optionally accepts an exit status
    rm -f $TEMP_FILE
    exit $1
}

function error_exit {

    # Display error message and exit
    echo "${PROGNAME}: ${1:-"Unknown Error"}" 1>&2
    clean_up 1
}

trap clean_up SIGHUP SIGINT SIGTERM

if [ $# != "1" ]; then
    usage
    error_exit "one file to print must be specified"
fi
if [ ! -f "$1" ]; then
    error_exit "file $1 cannot be read"
fi

lpr $1 > $TEMP_FILE || error_exit "cannot format file"
```

```
echo -n "Print file? [y/n]: "
read
if [ "$REPLY" = "y" ]; then
    lpr $TEMP_FILE || error_exit "cannot print file"
fi
clean_up
```

நிரல் விளக்கம்:

மேலே குறிப்பிட்ட முந்தைய நிரலின் நீட்சி இதுவாகும் . **function usage, function clean_up, function error_exit** ஆகிய துண்டு நிரல்கள் உள்ளன. இவை முறையே பயன்பாடு, கோப்பமைவு, பிழையறிதல் போன்றவற்றைத் தம்மகத்தே கொண்டுள்ளன. இந்நிரலின் மூலம் கோப்புத் தூய்மையாக்கம் செய்யப்பட்டு , புதிய கோப்பு அச்சிடப்படுகிறது.

கலைச்சொற்கள்:

தாடை – **signal**

கண்ணி – **trap command**

குறித்த நிகழ்செயல் – **particular action**

இடையினில் மறித்து – **interrupt in between**

தூய்மையாக்கம் – **house keeping**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 24
(getopts command) பொருத்தம் பெறும் கட்டளை

அளவுரு இருக்கும் அமைவினை அறிந்து
சரியான வண்ணம் சீர்மை செய்து
நிரலின் ஊடே நீந்திட வைத்து
வெளியிடு கிடைத்திட வித்திடும் பொருத்தமே.
- நிறற்பா 24

நிறற்பாவிளக்கம்:

இந்த பொருத்தம் பெறும் கட்டளையானது, அளவுரு அல்லது தருமதிப்பினை சரிபார்த்து அதைச் சரிவர நிரலில் பயன்படுத்த உதவுகிறது. இது பெரும்பாலும் **while loop** என்றழைக்கப்படக்கூடிய பொழுதெலாம் கட்டளையினூடே செலுத்தப் பயன்படுகிறது. **(This command is used to check valid command line argument are passed to script. Usually with while loop.)** இது பெரும்பாலும் தொடக்கப்பயனாளர்களால் பயன்படுத்தப்படுவதில்லை. நிரலின் ஊடே நீந்திட வைத்து என்பது, அளவுருவினை ஆராய்ந்து எதற்குப் பொருத்தமானது எது என்பதையும், தவறான தருமதிப்பைத் தவறு என்பதைக் காட்டுவதையும் குறிப்பிடுகிறது .

Syntax (பொது அமைவு**)**

இது கீழ்க்காணும் பொதுவமைவினைத் தன்னகத்தே கொண்டுள்ளது.

getopts {optsring} {variable1}

நிரல் **90**

```
#!/bin/bash
#go.sh
#niral90.sh
while getopts ":a" opt; do
  case $opt in
    a)
```

```
      echo "-a was triggered!" >&2
      ;;
    \?)
      echo "Invalid option: -$OPTARG" >&2
      ;;
  esac
done
```

[illegible]:

[illegible].

[illegible]. (Calling it without any arguments)

```
# ./go_test.sh
```

[illegible] (Nothing happened? Right. getopts didn't see any valid or invalid options (letters preceded by a dash), so it wasn't triggered.)

[illegible]. (Calling it with non-option arguments)

```
# ./go_test.sh /etc/passwd
```

[illegible]. (Again — nothing happened. The very same case: getopts didn't see any valid or invalid options (letters preceded by a dash), so it wasn't triggered.)

The arguments given to your script are of course accessible as $1 - ${N}.

Calling it with option-arguments

Now let's trigger getopts: Provide options.

[illegible].

```
# ./go_test.sh -b
```

Invalid option: -b

[illegible]

As expected, getopts didn't accept this option and acted like told above: It placed? into $opt and the invalid option character (b) into $OPTARG. With our case statement, we were able to detect this.

Now, a valid one (-a):

[illegible]

```
# ./go_test.sh -a
-a was triggered!
```

You see, the detection works perfectly. The a was put into the variable $opt for our case statement.

Of course it's possible to mix valid and invalid options when calling:

```
# ./go_test.sh -a -x -b -c
-a was triggered!
Invalid option: -x
Invalid option: -b
Invalid option: -c
```

[illegible]

```
# ./go_test.sh -a -a -a -a
-a was triggered!
-a was triggered!
-a was triggered!
-a was triggered!
```

[illegible] 91

```
 #!/bin/bash
#niral91.sh
# Usage: ani -n -a -s -w -d
# help_ani() To print help
```

```
help_ani()
{
  echo "Usage: $0 -n -a -s -w -d"
  echo "Options: These are optional argument"
  echo " -n name of animal"
  echo " -a age of animal"
  echo " -s sex of animal "
  echo " -w weight of animal"
  echo " -d demo values (if any of the above options are used "
  echo " their values are not taken)"
  exit 1
}
#
#Set default value for variable
#
isdef=0
na=Moti
age="2 Months" # may be 60 days, as U like it!
sex=Male
weight=3Kg
#
#if no argument
#
if [ $# -lt 1 ]; then
  help_ani
fi
while getopts n:a:s:w:d opt
do
  case "$opt" in
    n) na="$OPTARG";;
    a) age="$OPTARG";;
    s) sex="$OPTARG";;
```

```
    w) weight="$OPTARG";;
    d) isdef=1;;
    \?) help_ani;;
  esac
done
if [ $isdef -eq 0 ]
then
   echo "Animal Name: $na, Age: $age, Sex: $sex, Weight: $weight (user define mode)"
else
  na="Pluto Dog"
  age=3
  sex=Male
  weight=20kg
   echo "Animal Name: $na, Age: $age, Sex: $sex, Weight: $weight (demo mode)"
fi
```

[illegible]:

[illegible], [illegible], [illegible], [illegible](demo) [illegible].

We have script called ani which has syntax as

ani -n -a -s -w -d

Options: These are optional argument

-n name of animal

-a age of animal

-s sex of animal

-w weight of animal

-d demo values (if any of the above options are used their values are not taken)

[illegible]:

Save it and run as follows

 # chmod +x ani

 # ani -n Lassie -a 4 -s Female -w 20Kg

 # ani -a 4 -s Female -n Lassie -w 20Kg

 # ani -n Lassie -s Female -w 20Kg -a 4

 # ani -w 20Kg -s Female -n Lassie -a 4

 # ani -w 20Kg -s Female

 # ani -n Lassie -a 4

 # ani -n Lassie

 # ani -a 2

வெளியீட்டு விளக்கம்:

இங்கு உற்றுப் பார்க்க வேண்டியது என்னவென்றால், நாம் தருமதிப்பை பல்வேறு வண்ணங்களில் மாற்றித் தரலாம். அவற்றை பொருத்தம் பெறும் கட்டளை நிரலுக்குத் தக்கவாறு மாற்றி விடையறிய உதவுகிறது. கீழே தருமதிப்புக்கள் (அளவுருக்கள்) மாற்றி மாற்றிக் கொடுக்கப்பட்டுள்ளன. ஆனால் விடை எவ்வாறு வருகிறது என்பதைக் காணவும்.

 # ani -nLassie -a4 -sFemal -w20Kg

இங்கு **space** எதுவும் **option** களுக்கும் மதிப்புக்களுக்கும் நடுவே கொடுக்கப்படவில்லை. ஆனாலும் விடையானது,அருமையாக வருகிறது.

 # ani -nLassie-a4-sFemal-w20Kg

 # ani -n Lassie -a 4 -s Female -w 20Kg -c Mammal

-c is not one of the valid options.

கலைச்சொற்கள்:

அளவுரு – **parameter**

அமைவு – **sequence system**

சரியான வண்ணம் சீர்மை – **check and validate**

நிரல் – **script**

வெளியிடு – **output or answer**

பொருத்தமே – **getopts command**

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் **– 25 (cut command)** வெட்டுக் கட்டளை

நீட்டமாய்த் தரவுகள் நெடுவரி நிற்க
குறித்த நிரைகள் கழற்றிட வேண்டினில்
வெட்டுக் கட்டளை வளைந்து கொடுக்க
வேண்டிய தீர்வு விரைந்து வருமே.
- நிரற்பா 25

நிரற்பாவிளக்கம்:

இயங்குதளத்தில் அல்லது கோப்பில் உள்ள தரவுகள், அடுக்கடுக்காக இருக்க, நமக்கு வேண்டிய நிரைகள் **(columns)** மட்டும் தனித்து எடுக்க வெட்டுக்கட்டளையினைப் பயன்படுத்தி தேவையான தீர்வினைப் பெறலாம்.

Syntax (பொது அமைவு**)**

வெட்டுக் கட்டளை கீழ்க்காணும் பொதுவமைவினைத் தன்னகத்தே கொண்டுள்ளது.

cut –d(delimiter) f(field no) filename

eg: cut –d: f1 /etc/passwd

நிரல் **91**

கீழ்க்காண்பவை ஒரு கோப்பில் **(data.txt)** இருப்பதாகக் கொள்க.

```
one   two   three four  five
alpha beta  gamma  delta epsilon
```

```
#!/bin/bash
#niral91.sh
#cut statement in the file filtering
cut -f 3 data.txt
```

நிரல் விளக்கம்:

இதில் இந்தக் கோப்பில் உள்ள மூன்றாம் நிரை மட்டும் வெளியீடாகக் கிடைக்கிறது.

three

gamma

நிரல் **92**

```
#!/bin/bash
#niral91.sh
#cut statement
cut -f 1-2,4-5 data.txt
```

நிரல் விளக்கம்:

இதில் ஒன்று, இரண்டு, நான்கு ஐந்து ஆகிய நிரைகள் வெளியீடாகக் கிடைக்கின்றன.

```
one    two    four   five
alpha beta   delta  epsilon
```

நிரல் **93**

```
#!/bin/bash
#niral93.sh
#cut statement example 3
cut -f 1 -d ':' /etc/passwd
```

நிரல் விளக்கம்:

இங்கு **delimiter (**எல்லை**)** என்று அழைக்கக் கூடிய ஒன்று பயன்படுத்தப்படுகிறது. இதில் : என்பதைக் கொண்டு நிரைகள் பிரிக்கப்பட்டு விடைகள் அதற்கேற்றாற்போன்று வருகின்றன. **f1** என்பது நிரை ஒன்றைக் குறிக்கிறது .

root

daemon

bin

sys

chope

```
TechFundaes
File  Edit  View  Terminal  Tabs  Help
[gatz@gbox]>cat matrixSpace.txt
name friends fav_apps timeline_enabled photos date_of_joining
neo 1252 give_hearts yes 151 05_JUN_03
morpheus 987 texas_poker no 245 12_APR_05
trinity 1562 angry_bird no 91 09_SEP_05
[gatz@gbox]>cut -d' ' -f-3 matrixSpace.txt
name friends fav_apps
neo 1252 give_hearts
morpheus 987 texas_poker
trinity 1562 angry_bird
[gatz@gbox]>
```

□□□□□ 94

#!/bin/bash

#niral94.sh

#delimiter example no 2

cut -f 1,3 -d ':' --output-delimiter=$'\t' /etc/passwd

□□□□□ □□□□□□□□:

□□□□□ □□□□□ □□□□□□□ □□□□□□□□ □□□□□□□ □□□□□□□□□□□□□□□□□□□□ □□□ tab □□□□□□□ □□□□□□ □□□□□□□□□□□ □□□□□□□□□□□□□□□□□.

root 0

daemon 1

bin 2

sys 3

chope 1000

□□□□□ 95

#!/bin/bash

#niral95.sh

#combine cut command with other unix command

ps axu | grep python | sed 's/\s\+/ /g' | cut -d' ' -f2,11-

நிரல் விளக்கம்:

பொதுவாக இவ்வகையான கட்டளைகள் எந்தவொரு கோப்பினையும் உடைத்து எளிய தரவுகளாக மாற்றிக் கையாள உதவுகிறது. இது அதிகமாக பொறி நிறைஞர்களால் பயன்படுத்தப்படுகிறது.

நிரல் வெளியீடு கீழ்வருமாறு அமைகிறது.

2231 /usr/bin/python /usr/lib/unity-lens-video/unity-lens-video

2311 /usr/bin/python /usr/lib/unity-scope-video-remote/unity-scope-video-remote

2414 /usr/bin/python /usr/lib/ubuntuone-client/ubuntuone-syncdaemon

2463 /usr/bin/python /usr/lib/system-service/system-service-d

3274 grep --color=auto python

கலைச்சொற்கள்:

தரவுகள் – **data**

நிரைகள் – **columns or fields**

வெட்டுக் கட்டளை – **cut command**

பொறி நிறைஞர் – **system admins**

எல்லை (வரம்புக்குறி) – **delimiter**

பிரிப்பான் - **separator**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 26
(paste command) ஒட்டுக் கட்டளை

ஒட்டுக் கட்டளை ஒற்றி எடுத்திட
வெவ்வேறு கோப்பில் உள்ள உரைகள்
தரவுகள் ஒன்றாய் தெள்ளி எடுத்து
வெளியிடாய் வந்திட வழிவகை செய்யுமே.
- நிரற்பா 26

நிரற்பாவிளக்கம்:

ஒட்டுக் கட்டளையினைக் கொண்டு கட்டளை வரி அமைத்தால், வெவ்வேறு கோப்புக்களில் உள்ள உரைகள், அவற்றில் உள்ள தரவுகளை எடுத்து ஒரே கோப்பில் உள்ள தரவு போன்று தர உதவுகிறது.

Syntax (பொது அமைவு**)**

paste [OPTION]... [FILE]...

Options are as follows:

-d, --delimiters=LIST reuse characters from LIST instead of tabs.

-s, --serial paste one file at a time instead of in parallel.

--help Display a help message, and exit.

--version Display version information, and exit.

paste file1.txt file2.txt

கீழ்க்காண்பவை நிரல்களாக அன்றி தனித்தனிக் கட்டளைகளாகக் கொடுக்கப்பட்டுள்ளன.

நிரல் **96**

#!/bin/bash

#niral96.sh

cat file1

Unix

Linux

Windows

cat file2

Dedicated server

Virtual server

cat file3

Hosting

Machine

Operating system

[illegible]:

[illegible].

[illegible] 97

#!/bin/bash

#niral97.sh

paste file1 file2

Unix   Dedicated server

Linux   Virtual server

Windows

paste file2 file1

Dedicated server Unix

Virtual server   Linux

Windows

நிரல் விளக்கம்:

இங்கு இரண்டு கோப்புகள் சேர்த்து ஒரே கோப்பாக மாற்றியமை விளக்கப்பட்டுள்ளது.

நிரல் **98**

```
#!/bin/bash
#niral98.sh
paste -d"|" file1 file2
Unix|Dedicated server
Linux|Virtual server
Windows|
```

நிரல் விளக்கம்:

இங்கு வரம்புக்குறி ஆக | குறியீடு பயன்படுத்தப்படுகிறது. எனவே தரவுகளிடையே | குறியீடானது பயன்படுத்தப்படுகிறது.

நிரல் **99**

```
#!/bin/bash
#niral99.sh
paste -s file1 file2
Unix    Linux   Windows
Dedicated server        Virtual server
```

நிரல் விளக்கம்:

இங்கு பொதுவாக ஒட்டுக்கட்டளையானது பயன்படுத்தப்படுகிறது.

நிரல் **100**

```
#!/bin/bash
#niral100.sh
paste -d"|," file1 file2 file3
Unix|Dedicated server,Hosting
Linux|Virtual server,Machine
Windows|,Operating system
cat file1 | paste - -
```

Unix Linux
Windows

நிரல் விளக்கம்:
இந்தக் கட்டளைத் தொடரில், மூன்று கோப்புக்கள் இணைந்து ஒரே வெளியீடாகக் கிடைக்கிறது.

கலைச்சொற்கள் :

ஒட்டுக் கட்டளை – paste command
தரவுகள் – data
உரைகள் – text or string
வெவ்வேறு கோப்பில் – different files
(கற்போம்)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -175- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 27
(tput command) இடு கட்டளை

முனையம் திரையளவு மொத்தமாய் இருப்பினும்
சிறியதாய் இருப்பினும் செவ்வனே துப்புரவு
செய்திடல் நிரலில் சீர்மை கொண்டு
இடு கட்டளை இட்டிட வருமே.

\- நிரற்பா 27

நிரற்பாவிளக்கம்:

முனையம் என்றழைக்கப் படக்கூடிய **terminal** எந்த வகை அளவானாலும் **(maximize or minimize)** அதனை சீராகத் துப்புரவு செய்திட இடு **(tput)** கட்டளையானது உதவுகிறது.

Examples (எடுத்துக்காட்டுக்கள்)

tput longname

tput -T screen longname

tput colors

tput cols

tput bce && echo "True"

paste file1.txt file2.txt

நிரல் **101**

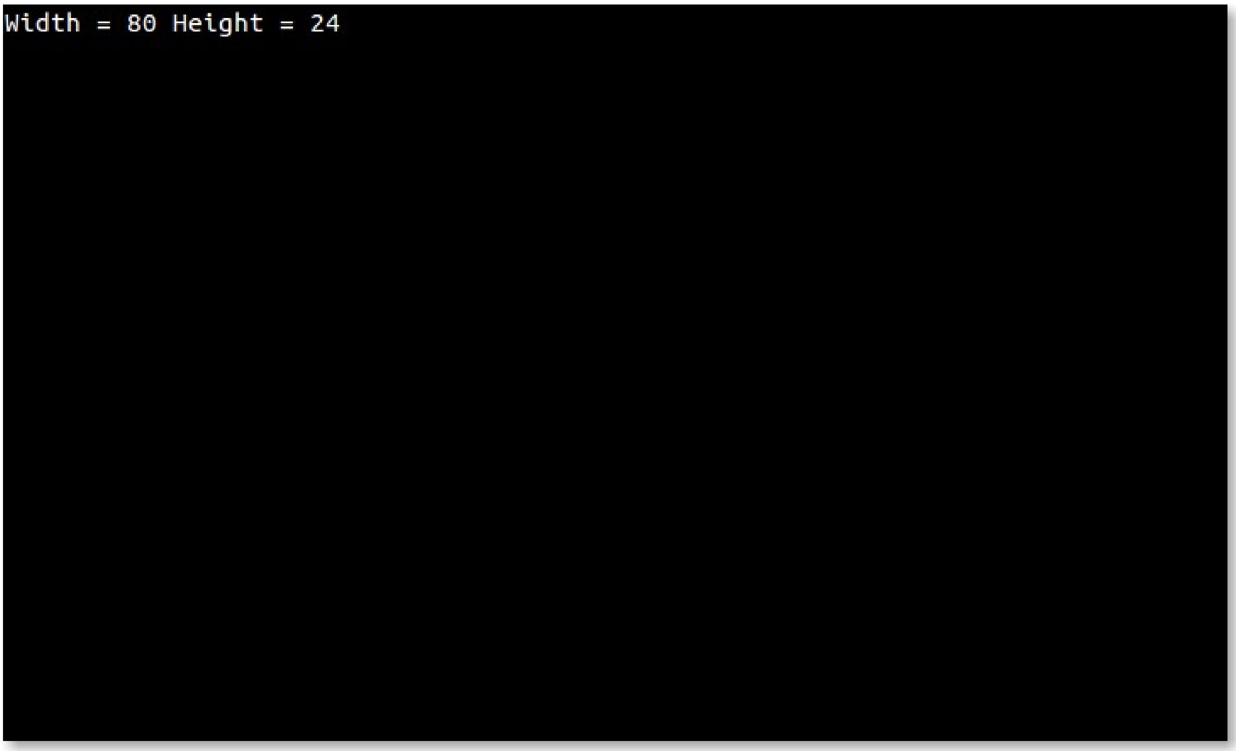
```
#!/bin/bash
#niral101.sh
alias term_size=`echo "Rows=$(tput lines) Cols=$(tput cols)"'
   # term_size2 - Dynamically display terminal window size
```

```
redraw() {
    clear
    echo "Width = $(tput cols) Height = $(tput lines)"
}



trap redraw WINCH

redraw
while true; do
    :
done
```

நிரல் விளக்கம்**:**

இங்கு **SIGWINCH** என்ற தாடையானது **(signal)** கொடுக்கப்பட்டு கீழ்க்காணுமாறு திரையினை உருமாற்றம் செய்யப்பயன்படுகிறது**.**

நிரல் வெளியீடு**:**



சுட்டி கட்டுப்பாடு **(Controlling the Cursor)**

Capname Description

sc Save the cursor position

rc Restore the cursor position

home Move the cursor to upper left corner (0,0)

cup <row> <col> Move the cursor to position row, col

cud1 Move the cursor down 1 line

cuu1 Move the cursor up 1 line

civis Set to cursor to be invisible

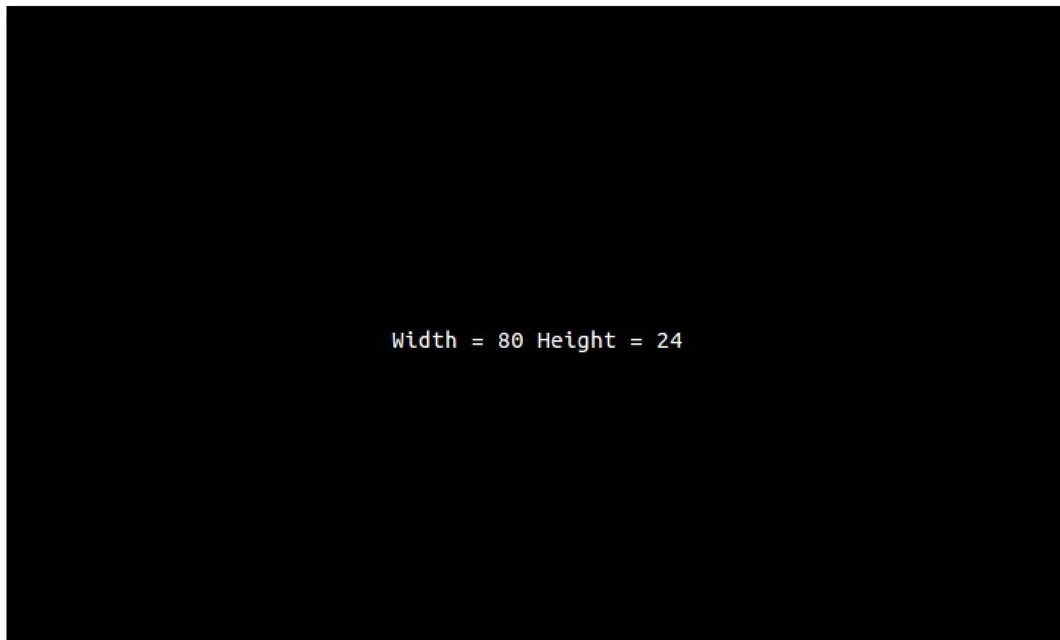
cnorm Set the cursor to its normal state

□□□□□ 102

```
#!/bin/bash
#niral102.sh
   # term_size3 - Dynamically display terminal window size
   #              with text centering

   redraw() {
      local str width height length

      width=$(tput cols)
      height=$(tput lines)
      str="Width = $width Height = $height"
      length=${#str}
      clear
      tput cup $((height / 2)) $(((width / 2) - (length / 2)))
      echo "$str"
   }

   trap redraw WINCH

   redraw
   while true; do
      :
```

```
    done
```

□□□□□ □□□□□□□□□:



□□□□□ □□□□□□□□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□ □□□□□□□□□□□□□ □□□□□□□□.

□□□ □□□□□□□□□□

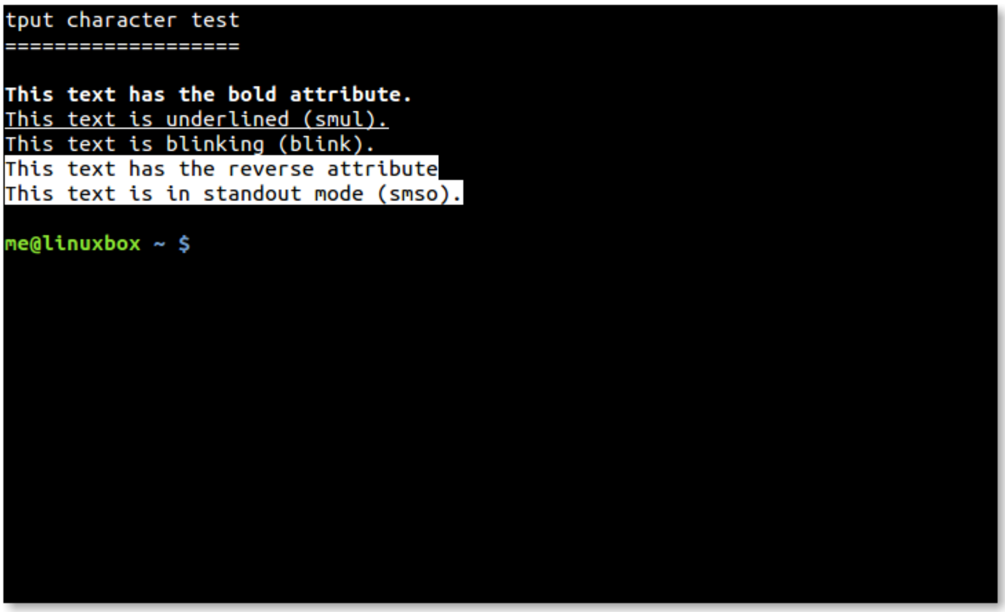
Capname	Description
bold	Start bold text
smul	Start underlined text
rmul	End underlined text
rev	Start reverse video
blink	Start blinking text
invis	Start invisible text
smso	Start "standout" mode
rmso	End "standout" mode
sgr0	Turn off all attributes
setaf <value>	Set foreground color
setab <value>	Set background color

□□□□□ 103

```
#!/bin/bash
#niral103.sh
    # tput_characters - Test various character attributes
```

```
clear

echo "tput character test"
echo "===================="
echo

tput bold;  echo "This text has the bold attribute.";     tput sgr0

tput smul;  echo "This text is underlined (smul).";       tput rmul

  # Most terminal emulators do not support blinking text (though
xterm
# does) because blinking text is considered to be in bad taste ;-)
tput blink; echo "This text is blinking (blink).";        tput sgr0

tput rev;   echo "This text has the reverse attribute";   tput sgr0

# Standout mode is reverse on many terminals, bold on others.
tput smso;  echo "This text is in standout mode (smso)."; tput rmso

tput sgr0
echo
```

நிரல் விளக்கம்:



இங்கு நிரலானது மேற்காணும் வண்ணம் அமைந்தவாறு இருக்கிறது. இயங்குதளங்களில் இது போன்று நிரல்கள் பயன்படுத்தப்பட்டு அழகு சேர்க்கப்படுகிறது . ரெட் ஹாட் இயங்குதளங்களைக் காட்டிலும் உபுண்டு போன்ற இயங்குதளங்களில் அதிகமாக இது போன்று நிரல்கள் பயன்படுத்தப்பட்டு அழகு சேர்க்கப்படுகின்றது .

கலைச்சொற்கள்:

இடு கட்டளை – **tput command**

எந்த வகை அளவானாலும் – **maximize or minimize**

உரைகள் – **text or string**

வெவ்வேறு கோப்பில் – **different files**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 28
(tput command) இடு கட்டளை

கட்டளை இடுவாம் கொண்டே வந்திடும்
வகைகள் பத்து வண்ணம் உண்டாம்
முனையம் சீருடன் மிளிர்ந்திட வேண்டி
நிறங்கள் பூட்டி நீட்டிடில் செய்யவே .

- நிரற்பா 28

நிரற்பாவிளக்கம்:

இடு கட்டளையானது தன்னகத்தே பத்து வண்ணங்களைக் கொண்டுள்ளது. அவற்றைக் கொண்டு அழகுற நமது முனையத்தின் வண்ணங்களை மாற்றி இன்புறலாம்.

உரை வண்ணங்கள் **(Text color)**

Value Color

0 Black

1 Red

2 Green

3 Yellow

4 Blue

5 Magenta

6 Cyan

7 White

8 Not used

9 Reset to default color

நிரல் **104**

```
#!/bin/bash
#script104.sh
   # tput_colors - Demonstrate color combinations.
```

```
for fg_color in {0..7}; do
    set_foreground=$(tput setaf $fg_color)
    for bg_color in {0..7}; do
        set_background=$(tput setab $bg_color)
        echo -n $set_background$set_foreground
        printf ' F:%s B:%s ' $fg_color $bg_color
    done
    echo $(tput sgr0)
done
```

[illegible]:

```
me@linuxbox ~ $ tput_colors
 F:0 B:0  F:0 B:1  F:0 B:2  F:0 B:3  F:0 B:4  F:0 B:5  F:0 B:6  F:0 B:7
 F:1 B:0  F:1 B:1  F:1 B:2  F:1 B:3  F:1 B:4  F:1 B:5  F:1 B:6  F:1 B:7
 F:2 B:0  F:2 B:1  F:2 B:2  F:2 B:3  F:2 B:4  F:2 B:5  F:2 B:6  F:2 B:7
 F:3 B:0  F:3 B:1  F:3 B:2  F:3 B:3  F:3 B:4  F:3 B:5  F:3 B:6  F:3 B:7
 F:4 B:0  F:4 B:1  F:4 B:2  F:4 B:3  F:4 B:4  F:4 B:5  F:4 B:6  F:4 B:7
 F:5 B:0  F:5 B:1  F:5 B:2  F:5 B:3  F:5 B:4  F:5 B:5  F:5 B:6  F:5 B:7
 F:6 B:0  F:6 B:1  F:6 B:2  F:6 B:3  F:6 B:4  F:6 B:5  F:6 B:6  F:6 B:7
 F:7 B:0  F:7 B:1  F:7 B:2  F:7 B:3  F:7 B:4  F:7 B:5  F:7 B:6  F:7 B:7
me@linuxbox ~ $
```

[illegible]:

[illegible].

[illegible] 105

[illegible] (Clearing the Screen)

[illegible].

CapnameDescription

smcup Save screen contents ([illegible] [illegible] [illegible].)

rmcup Restore screen contents ([illegible] [illegible] [illegible] [illegible].)

el Clear from the cursor to the end of the line ([illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].)

el1Clear from the cursor to the beginning of the line ([illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].)

ed Clear from the cursor to the end of the screen ([illegible] [illegible] [illegible] [illegible] [illegible] [illegible])

clear Clear the entire screen and home the cursor ([illegible] [illegible] [illegible] [illegible].)

```
#!/bin/bash
#niral105.sh
   # tput_menu: a menu driven system information program

   BG_BLUE="$(tput setab 4)"
   BG_BLACK="$(tput setab 0)"
   FG_GREEN="$(tput setaf 2)"
   FG_WHITE="$(tput setaf 7)"

   # Save screen
   tput smcup

   # Display menu until selection == 0
   while [[ $REPLY != 0 ]]; do
     echo -n ${BG_BLUE}${FG_WHITE}
     clear
     cat <<- _EOF_
       Please Select:

       1. Display Hostname and Uptime
```

```
  2. Display Disk Space
  3. Display Home Space Utilization
  0. Quit

_EOF_

 read -p "Enter selection [0-3] > " selection

 # Clear area beneath menu
 tput cup 10 0
 echo -n ${BG_BLACK}${FG_GREEN}
 tput ed
 tput cup 11 0

 # Act on selection
 case $selection in
  1)  echo "Hostname: $HOSTNAME"
     uptime
     ;;
  2)  df -h
     ;;
  3)  if [[ $(id -u) -eq 0 ]]; then
       echo "Home Space Utilization (All Users)"
       du -sh /home/* 2> /dev/null
     else
       echo "Home Space Utilization ($USER)"
       du -s $HOME/* 2> /dev/null | sort -nr
     fi
     ;;
  0)  break
     ;;
  *)  echo "Invalid entry."
```

```
        ;;
    esac
    printf "\n\nPress any key to continue."
    read -n 1
  done

  # Restore screen
  tput rmcup
  echo "Program terminated."
```

[illegible] [illegible]:

```
Please Select:

1. Display Hostname and Uptime
2. Display Disk Space
3. Display Home Space Utilization
0. Quit

Enter selection [0-3] > 2


Filesystem      Size  Used Avail Use% Mounted on
rootfs          5.8G  4.0G  1.6G  73% /
/dev/root       5.8G  4.0G  1.6G  73% /
devtmpfs        215M     0  215M   0% /dev
tmpfs            44M  260K   44M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            88M     0   88M   0% /run/shm
/dev/mmcblk0p5   60M  9.6M   50M  17% /boot


Press any key to continue.
```

[illegible]. [illegible], [illegible].

கலைச்சொற்கள்:

இடுவாம் – **tput command**

முனையம் – **terminal**

நிறங்கள் – **colors**

வண்ணங்கள் **- colors**

அழித்தல் **- clearing**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 29
(tput command) இடு கட்டளை தொடர்ச்சி

இடுவாம் கட்டளை ஈண்டெடுத் தியம்பிடும்
இந்நிரல் கொண்டு எடுத்தாண் டிடவே
கீழ்வரும் விதமாய் கடிகை வெளியிடு
பெரிதாய் வந்து பண்புரு செய்யுமே.
- நிரற்பா 29

நிரற்பாவிளக்கம்:

tput என்னும் இடு கட்டளை கொண்டு கீழ்க்காணும் நிரலினைச் சரிவர எழுதி நிறைவு செய்தால், அழகான மணிப்பொறி **(terminal clock)** பண்புரு **(tensor)** அமைவில் வந்து வெளியீடாகக் **(output)** கிடைக்கும்.

நிரல் **106**

```
#niral106
#!/bin/bash

  # tclock - Display a clock in a terminal

  BG_BLUE="$(tput setab 4)"
  FG_BLACK="$(tput setaf 0)"
  FG_WHITE="$(tput setaf 7)"

  terminal_size() { # Calculate the size of the terminal

    terminal_cols="$(tput cols)"
    terminal_rows="$(tput lines)"
  }
```

```
banner_size() {

  # Because there are different versions of banner, we need to
  # calculate the size of our banner's output

  banner_cols=0
  banner_rows=0

  while read; do
    [[ ${#REPLY} -gt $banner_cols ]] && banner_cols=${#REPLY}
    ((++banner_rows))
  done < <(banner "12:34 PM")
}

display_clock() {

  # Since we are putting the clock in the center of the terminal,
  # we need to read each line of banner's output and place it in the
  # right spot.

  local row=$clock_row

  while read; do
    tput cup $row $clock_col
    echo -n "$REPLY"
    ((++row))
  done < <(banner "$(date +'%I:%M %p')")
}

# Set a trap to restore terminal on Ctrl-c (exit).
# Reset character attributes, make cursor visible, and restore
# previous screen contents (if possible).
```

```
trap 'tput sgr0; tput cnorm; tput rmcup || clear; exit 0' SIGINT

# Save screen contents and make cursor invisible
tput smcup; tput civis

# Calculate sizes and positions
terminal_size
banner_size
clock_row=$(((terminal_rows - banner_rows) / 2))
clock_col=$(((terminal_cols - banner_cols) / 2))
progress_row=$((clock_row + banner_rows + 1))
progress_col=$(((terminal_cols - 60) / 2))

# In case the terminal cannot paint the screen with a background
# color (tmux has this problem), create a screen-size string of
# spaces so we can paint the screen the hard way.

blank_screen=
for ((i=0; i < (terminal_cols * terminal_rows); ++i)); do
  blank_screen="${blank_screen} "
done

# Set the foreground and background colors and go!
echo -n ${BG_BLUE}${FG_WHITE}
while true; do

  # Set the background and draw the clock

    if tput bce; then # Paint the screen the easy way if bce is supported
      clear
```

```
else # Do it the hard way
  tput home
  echo -n "$blank_screen"
fi
tput cup $clock_row $clock_col
display_clock

# Draw a black progress bar then fill it in white
tput cup $progress_row $progress_col
echo -n ${FG_BLACK}
echo -n "############################################################"
tput cup $progress_row $progress_col
echo -n ${FG_WHITE}

# Advance the progress bar every second until a minute is used up
for ((i = $(date +%S);i < 60; ++i)); do
  echo -n "#"
  sleep 1
done
done
```

நிரல் விளக்கம்:

மேற்காணும் நிரலில் **terminal clock** என்றழைக்கப்படக்கூடிய முனையக் கடிகையானது அமைக்கப்பட்டுள்ளது. இங்கு பின்புலம் நீலமாகவும், முன்புலம் வெள்ளையாகவும் நிலை உணர்த்திப்பட்டை **(progress bar)** கறுமை நிறமாகவும் அமைக்கப்பட்டுள்ளது. இதில் கடைசியில் வரும் ஆகக் கட்டளையானது **(for loop) progress bar** ஐ ஒவ்வொரு நொடிக்கும் ஒன்று என்ற வண்ணம் நிறமாற்றி பயனருக்கு அளிக்கிறது.

நிரல் வெளியீடு:

□□□□□□□□□□□□□□:

□□□□□□□□ □□□□□□□ – tput command

□□□□□□ – script

□□□□□□ (□□□□□□□□□□) – clock

□□□□□□□□□ - output

□□□□□□□□ – tensor

□□□□□ □□□□□□□□□□□□□□□□□□ - progress bar

□□□□□□□□ □□□□□□ - terminal clock

□□□□□ □□□□□□□ – for loop

(□□□□□□□□□)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 30 (nohup command) செயலிழக்காக் கட்டளை

பயனர் தந்த பின்புலக் கட்டளை
முனையம் கொண்டு மடிந்த பின்னரும் தன்செயல் செய்து தீர்வு விடுக்க
செயலி ழக்கா சீர்வரி உதவுமே
- நிரற்பா 30

நிரற்பாவிளக்கம் :

பயனர் கொடுத்த பின்புலத்தில் செயல்படும் கட்டளையானது, குறிப்பிட்ட முனையமானது, மூடப்பட்ட பின்னரும் செயல்பட வேண்டுமெனில் செயலிழக்கா (nohup) கட்டளை பயன்படுத்தப்படல் வேண்டும்.
பொது அமைவு (General syntax):
#nohup command-name &
#nohup /path/to/command-name arg1 arg2 &
பின்புலத்தில் இருக்கும் வேலைகளைப் பார்க்ககீழ்க்காணும் கட்டளை உதவுகிறது.
#jobs –l
நிரல் 107
#niral107
#!/bin/bash
nohup find / -xdev -type f -perm +u=s -print > out.txt &
நிரல் விளக்கம்:
குறிப்பிட்ட நிரலானது பின்புலத்தில் எவ்வாறு find கட்டளையினை செயலிழக்காமல் செய்யும் வண்ணம் பயன்படுத்துவது என்று விளக்குகிறது. நிரல் 108
#niral108
#!/bin/bash

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -193- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

Example: Printing lines to both standard output & standard error

```
while(true)
do
echo "standard output"
echo "standard error" 1>&2
sleep 1;
done
```

□□□□□□ □□□□□□□□□ 1:

Execute the script without redirection

```
$ nohup sh custom-script.sh &
[1] 12034
$ nohup: ignoring input and appending output to `nohup.out'
$ tail -f nohup.out
standard output
standard error
standard output
standard error
..
```

□□□□□□ □□□□□□□□□ 2:

Execute the script with redirection

```
$ nohup sh custom-script.sh > custom-out.log &
[1] 11069
$ nohup: ignoring input and redirecting stderr to stdout
```

□□□□□□ □□□□□□□□□ 3:

```
$ tail -f custom-out.log
standard output
standard error
standard output
standard error
..
```

If you log-out of the shell and login again, you'll still see the custom-script.sh running in the background.

Tip

Term

- Need to run command in background with terminal detached?
- Use screen(1) & nohup(1)
  - screen - screen manager with VT100/ANSI terminal emulation
  - nohup - run a command immune to hangups, with output to a non-tty
  - Difference – while nohup is detaching command from terminal and redirecting output to file, screen is detaching itself (not command) from terminal.

[illegible] 4:

$ ps aux | grep prasanna

prasanna 12034 0.0 0.1 4912 1080 pts/2 S 14:10 0:00 sh custom-script.sh

nohup command with password less authentication:

[illegible] (passwordless authentication) [illegible], [illegible] (server) [illegible].

[illegible], [illegible].

$ nohup scp file_to_copy user@server:/path/to/copy/the/file > nohup.out 2>&1

[illegible], [illegible], [illegible] nohup [illegible].

கலைச்சொற்கள்:

பயனர் – **user**

பின்புலக் கட்டளை – **background command**

முனையம் – **terminal**

மடிந்த பின்னரும் – **closed after the terminal**

தீர்வு விடுக்க – **give output**

செயலி ழக்கா – **nohup command**

கடவுச்சொல்லற்ற நுழைவமைவு – **passwordless authentication**

வழங்கி - **Server**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 31

பொழுதொடு சேரும் புதிய பதிகை
தவறிலா விதமாய்த் துப்புரவு செய்திட
பொருள் கொண்டு பின்வரும் சீரிய
நிரலின் அமைவு நீடித்து உதவுமே.
- நிரற்பா 31

நிரற்பாவிளக்கம்:

இங்கு அமைந்துள்ள விதவிதமான நிரல்கள் எவ்வாறு தாமாகவே ஏற்படும் பதிகைகள் அழிக்கப்பயன்படுகின்றன என்பதைக் காட்டிட உதவுகின்றன .

பொது அமைவு **(General syntax):**

நிரல் **109**

```
#niral109
# Cleanup
# Run as root, of course.

cd /var/log
cat /dev/null > messages
cat /dev/null > wtmp
echo "Log files cleaned up."
```

நிரல் விளக்கம்**:**

இது மிகவும் எளிமையான நிரல். இதனை நாம் வேர்ப்பயனர் என்னும் பயனரைக் கொண்டு **(root user)** மட்டுமே இயக்க முடியும் . பொதுவான பயனர்கள் இயக்க வேண்டுமெனில் அதற்குத் தனியாக நாம் உத்தரவுகளைக் **(permissions)** கொடுக்க வேண்டும்.

நிரல் **110**

```
#niral110
#!/bin/bash
# Proper header for a Bash script.
```

```
# Cleanup, version 2

# Run as root, of course.
# Insert code here to print error message and exit if not root.

LOG_DIR=/var/log
# Variables are better than hard-coded values.
cd $LOG_DIR

cat /dev/null > messages
cat /dev/null > wtmp


echo "Logs cleaned up."

exit #  The right and proper method of "exiting" from a script.
     #  A bare "exit" (no parameter) returns the exit status
     #+ of the preceding command.
```

[illegible] [illegible]:

[illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] (/var/log) [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] (reset) [illegible].

```
#niral111
#!/bin/bash
# Cleanup, version 3

#  Warning:
#  -------
#  This script uses quite a number of features that will be explained
```

```
#+ later on.
#  By the time you've finished the first half of the book,
#+ there should be nothing mysterious about it.

LOG_DIR=/var/log
ROOT_UID=0     # Only users with $UID 0 have root privileges.
LINES=50       # Default number of lines saved.
E_XCD=86       # Can't change directory?
E_NOTROOT=87   # Non-root exit error.

# Run as root, of course.
if [ "$UID" -ne "$ROOT_UID" ]
then
  echo "Must be root to run this script."
  exit $E_NOTROOT
fi

if [ -n "$1" ]
# Test whether command-line argument is present (non-empty).
then
  lines=$1
else
  lines=$LINES # Default, if not specified on command-line.
fi

#  Stephane Chazelas suggests the following,
#+ as a better way of checking command-line arguments,
#+ but this is still a bit advanced for this stage of the tutorial.
#
#    E_WRONGARGS=85  # Non-numerical argument (bad argument
format).
#
```

```
#    case "$1" in
#    ""      ) lines=50;;
#    *[!0-9]*) echo "Usage: `basename $0` lines-to-cleanup";
#     exit $E_WRONGARGS;;
#    *       ) lines=$1;;
#    esac
#
#* Skip ahead to "Loops" chapter to decipher all this.


cd $LOG_DIR

if [ `pwd` != "$LOG_DIR" ]  # or   if [ "$PWD" != "$LOG_DIR" ]
                            # Not in /var/log?
then
  echo "Can't change to $LOG_DIR."
  exit $E_XCD
fi  # Doublecheck if in right directory before messing with log file.

# Far more efficient is:
#
# cd /var/log || {
#   echo "Cannot change to necessary directory." >&2
#   exit $E_XCD;
# }

tail -n $lines messages > mesg.temp # Save last section of message
log file.
mv mesg.temp messages               # Rename it as system log file.

#  cat /dev/null > messages
#* No longer needed, as the above method is safer.
```

```
cat /dev/null > wtmp  #  ': > wtmp' and '> wtmp'  have the same effect.
echo "Log files cleaned up."
#  Note that there are other log files in /var/log not affected
#+ by this script.

exit 0
#  A zero return value from the script upon exit indicates success
#+ to the shell.
```

நிரல் விளக்கம்:

இது மற்ற நிரல்கள் போல் பொதுவாக இயங்காமல், எந்தப் பயனர் இயக்குகிறார் அவருக்கு எவ்வகையான உத்தரவுகள் வழங்கப்பட்டுள்ளன என்பதையெல்லாம் ஆய்ந்தறிந்து அதற்கேற்றாற்போன்று இயங்கி வெளியீட்டினைத் தருகிறது. பொதுவாக இவ்வகையான நிரல்களை இயக்கி எழுதும் பொழுது நாம் பயனரையும் அவருடைய உத்தரவுகளையும் கருத்தில் கொண்டு செயல்படுவது தேவையாகிறது.

கலைச்சொற்கள்:

பயனர் – **user**

புதிய பதிகை – **new logs**

தவறிலா விதமாய் – **right manner**

துப்புரவு செய்திட – **clean**

வேர்ப்பயனர் – **root user**

மீட்டமை – **reset**

அமைவுகள் **- settings**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 32

உள்ளிடு வெளியிடு உற்று நோக்கி
வேண்டிய வைத்து வேறதைத் திருப்பி
பொதியதன் தன்மை பொருத்தி பண்பாய்
பேணிட பயன்படும் பின்வரும் நிரலே.
- நிறற்பா 32

நிரற்பாவிளக்கம்:

நிரலில் கொடுக்கப்படும் உள்ளீடு மற்றும் வெளியீடுகளை உற்று கவனித்து, தேவையானதை வைத்து மற்றதைப் பிரித்து, பொதி எனப்படுகின்ற **package** சரிவர பயன்படுத்திட உதவும் நிரலே கீழ்க்காணும் நிரலாகும்.

நிரல் **112**

```
#!/bin/bash
#niral112
# rpm-check.sh

#  Queries an rpm file for description, listing,
#+ and whether it can be installed.
#  Saves output to a file.
#
#  This script illustrates using a code block.

SUCCESS=0
E_NOARGS=65

if [ -z "$1" ]
then
  echo "Usage: `basename $0` rpm-file"
  exit $E_NOARGS
```

```
fi

{ # Begin code block.
  echo
  echo "Archive Description:"
  rpm -qpi $1       # Query description.
  echo
  echo "Archive Listing:"
  rpm -qpl $1       # Query listing.
  echo
  rpm -i --test $1  # Query whether rpm file can be installed.
  if [ "$?" -eq $SUCCESS ]
  then
    echo "$1 can be installed."
  else
    echo "$1 cannot be installed."
  fi
  echo              # End code block.
} > "$1.test"       # Redirects output of everything in block to file.

echo "Results of rpm test in file $1.test"

# See rpm man page for explanation of options.

exit 0
```

[illegible]:

[illegible], Archive Description [illegible], [illegible] Archive Listing [illegible]. [illegible] SUCCESS [illegible] rpm can be installed [illegible], [illegible] cannot be installed. [illegible]. [illegible]

முடிகிறது. இது பொதுவாக அனைத்து வகையான பொதிகளுக்கும் உதவும் வகையில் அமைக்கப்பட்டுள்ளது.

நிரல் **113 -** சிறு விவர நிரல் **(tips script)**

```
#!/bin/bash
#niral113

# uppercase.sh : Changes input to uppercase.

tr 'a-z' 'A-Z'
#  Letter ranges must be quoted
#+ to prevent filename generation from single-letter filenames.
Exit 0
```

நிரல் விளக்கம்**:**

இது ஓர் எளிமையான நிரலாகும். கொடுக்கப்படும் எதையும் சிறிய எழுத்திலிருந்து பெரிய எழுத்திற்கு மாற்ற உதவும் நிரல். **tr** என்னும் கட்டளை **translate** என்னும் பொருள்படும்படி அமைக்கப்பட்டுள்ளது. இதற்கு **uppercase.sh** என்ற பெயர் இடப்பட்டுள்ளது**.** பொதுவாக **ls –l** என்று கொடுத்தால் ஒரு அடைவிற்குள் இருக்கும் அனைத்து கோப்புகளும் வெளியீடாகக் கிடைக்கும். ஆனால் இங்கு இரண்டையும் சேர்த்து **ls -l | ./uppercase.sh** கொடுப்பதால், பொதுவாகக் கிடைக்கும் வெளியீடானது பெரிய எழுத்துக்களாக மாற்றப்பட்டுக் கிடைக்கிறது.

நிரல் வெளியீடு**:**

```
bash$ ls -l | ./uppercase.sh
-RW-RW-R--    1 BOZO  BOZO       109 APR  7 19:49 1.TXT
-RW-RW-R--    1 BOZO  BOZO       109 APR 14 16:48 2.TXT
-RW-R--R--    1 BOZO  BOZO       725 APR 20 20:56 DATA-FILE
```

நிரல் **114 (running a loop in background -** வளைவுக் கட்டளையைப் பின்புலத்தில் இயக்குதல்**.)**

```
#!/bin/bash
```

```
#niral114
# background-loop.sh

for i in 1 2 3 4 5 6 7 8 9 10          # First loop.
do
  echo -n "$i "
done & # Run this loop in background.
       # Will sometimes execute after second loop.

echo   # This 'echo' sometimes will not display.

for i in 11 12 13 14 15 16 17 18 19 20   # Second loop.
do
  echo -n "$i "
done

echo   # This 'echo' sometimes will not display.

#
=====================================================
=======
```

நிரல் விளக்கம்:

மேற்கண்ட நிரல் தன்னகத்தே உள்ள வளைவுக் கட்டளையினை பின்புலத்தில் இயக்கப் பயன்படுகிறது. இங்கு & என்னும் குறியீடானது, குறிப்பிட்ட வளைவுக்கட்டளையினை **(loop)** பின்புலத்தே **(background)** இயங்க உதவுகிறது. இங்கு உற்று நோக்க வேண்டியது என்னவெனில், ஒவ்வொரு முறையும் வெவேறு வெளியீடுகள் கிடைக்கிறது. எடுத்துக்காட்டிற்காக , சில நிரல் வெளியீடுகள் தரப்பட்டுள்ளன.

நிரல் வெளியீடு:

இங்கு ஐந்து வகையான வெளியீடுகள் நிரலினை வெவ்வேறு நேரங்களில் இயக்கும் பொழுது கிடைக்குமென்று கொடுக்கப்பட்டுள்ளது.

 # The expected output from the script:

```
# 1 2 3 4 5 6 7 8 9 10
# 11 12 13 14 15 16 17 18 19 20

# Sometimes, though, you get:
# 11 12 13 14 15 16 17 18 19 20
# 1 2 3 4 5 6 7 8 9 10 bozo $
# (The second 'echo' doesn't execute. Why?)

# Occasionally also:
# 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
# (The first 'echo' doesn't execute. Why?)

# Very rarely something like:
# 11 12 13 1 2 3 4 5 6 7 8 9 10 14 15 16 17 18 19 20
# The foreground loop preempts the background one.

exit 0
```

[illegible]:

[illegible] – input

[illegible] – output

[illegible] [illegible] – anlysing

[illegible] – required

[illegible] – non-required

[illegible] - package

[illegible] - script

[illegible] – RPM package (Red Hat Package Manager)

[illegible] – loop

[illegible] - background

([illegible])

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 33

மற்ற கணினி மொழிகள் போலிலா
முழுவெண் கோவை முடிக்க வகையிலா
இடமும் பொருளும் எடுத்து இயங்கிடும்
சூழல் முனையம் சார்ந்த மாறியே.
- நிரற்பா 33

நிரற்பாவிளக்கம்:

மற்ற கணினி உயர் நிலை மொழிகள் போலில்லாமல், முனையக் குறுநிரலானது **(shell script)** முழுஎண் **(integer),** சரம் அல்லது கோவை **(string)**ஆகியவற்றை அறுதியிட்டு கூறாமல் அவை அவை இருக்கும் இடத்திற்கேற்ப பொருளை இயங்கு நேரத்தில் **(run time)** எடுத்துக் கொண்டு முனையத்தின் சூழலைச் **(depends on the terminal)** சார்ந்து அமைவதே மாறியாகும் **(variable).**

நிரல் **115**

```
#!/bin/bash
#niral115.sh
# int-or-string.sh

a=2334                  # Integer.
let "a += 1"
echo "a = $a "           # a = 2335
echo                    # Integer, still.


b=${a/23/BB}             # Substitute "BB" for "23".
                         # This transforms $b into a string.
echo "b = $b"            # b = BB35
declare -i b             # Declaring it an integer doesn't help.
echo "b = $b"            # b = BB35
```

```
let "b += 1"            # BB35 + 1
echo "b = $b"           # b = 1
echo                    # Bash sets the "integer value" of a string to 0.

c=BB34
echo "c = $c"           # c = BB34
d=${c/BB/23}            # Substitute "23" for "BB".
                        # This makes $d an integer.
echo "d = $d"           # d = 2334
let "d += 1"            # 2334 + 1
echo "d = $d"           # d = 2335
echo


# What about null variables?
e=''                    # ... Or e="" ... Or e=
echo "e = $e"           # e =
let "e += 1"            # Arithmetic operations allowed on a null variable?
echo "e = $e"           # e = 1
echo                    # Null variable transformed into an integer.

# What about undeclared variables?
echo "f = $f"           # f =
let "f += 1"            # Arithmetic operations allowed?
echo "f = $f"           # f = 1
echo                    # Undeclared variable transformed into an integer.
#
# However ...
let "f /= $undecl_var"   # Divide by zero?
#   let: f /= : syntax error: operand expected (error token is " ")
# Syntax error! Variable $undecl_var is not set to zero here!
```

```
#
# But still ...
let "f /= 0"
#   let: f /= 0: division by 0 (error token is "0")
# Expected behavior.


#  Bash (usually) sets the "integer value" of null to zero
#+ when performing an arithmetic operation.
#  But, don't try this at home, folks!
#  It's undocumented and probably non-portable behavior.


# Conclusion: Variables in Bash are untyped,
#+ with all attendant consequences.

exit $?
```

நிரல் விளக்கம்:

ஆங்கிலத்திலே **untyped** என்றழைக்கப்படக்கூடிய அறிவிப்புச் செய்யப்படாத மாறிகள், அவைகளின் அமைவிடத்தைப் பொறுத்து, ஆசி அல்லது சாபம் ஆகிய இரண்டையுமே தருகின்றன. குறுநிரலில் எளிதாக குறியீட்டு வரிகளைக் கொடுக்க, அல்லது நீங்கள் செயலிழக்கப் போதுமான கயிறு கொடுக்க இத்தகைய அறிவிக்கப்படாத மாறிகள் உதவுகின்றன. எனினும், இவை துல்லியமற்ற குறுநிரல்களை **(non-accurate scripts)** எழுதிப்பார்க்க , நுட்பமான பிழைகளுக்கு உத்தரவு கொடுக்க உதவுகின்றன. மாறிகளை தன்னகத்தே வைத்துக் கொண்டு, குறுநிரலினை தேவையான போது இயக்கம் செய்வதை முனையம் பொதுவாக மறுக்கிறது.

நிரல் **116:**

```
#!/bin/bash
# niral116.sh

# Does a 'whois domain-name' lookup on any of 3 alternate servers:
```

```
#                    ripe.net, cw.net, radb.net

# Place this script -- renamed 'wh' -- in /usr/local/bin

# Requires symbolic links:
# ln -s /usr/local/bin/wh /usr/local/bin/wh-ripe
# ln -s /usr/local/bin/wh /usr/local/bin/wh-apnic
# ln -s /usr/local/bin/wh /usr/local/bin/wh-tucows

E_NOARGS=75



if [ -z "$1" ]
then
  echo "Usage: `basename $0` [domain-name]"
  exit $E_NOARGS
fi

# Check script name and call proper server.
case `basename $0` in    # Or:    case ${0##*/} in
    "wh"       ) whois $1@whois.tucows.com;;
    "wh-ripe"  ) whois $1@whois.ripe.net;;
    "wh-apnic" ) whois $1@whois.apnic.net;;
    "wh-cw"    ) whois $1@whois.cw.net;;
    *          ) echo "Usage: `basename $0` [domain-name]";;
esac

exit $?
```

[illegible]:

[illegible], [illegible], [illegible]. [illegible]. [illegible] **ripe.net,**

cw.net, radb.net ஆகிய தளங்களில் இயக்குகிறோம். குறிப்பிட்ட தளங்கள் சரியான பெயர் கொண்டவையா என்பதைக் கண்டுகொண்டு அதை வெளியீடாகத் தருகிறது.

நிரல் **117:**

கீழே ஒரே நிரலாகக் கொடுக்கப்படாமல், தனித்தனி கட்டளைவரிகளாகக் கொடுக்கப்பட்டுள்ளது. அனைத்தும் எளிமையான கட்டளைவரிகள்தாம். எவ்வாறு **echo** என்ற கட்டளையானது வெவ்வேறு வகையாக பயன்படுத்தப்படுகிறது என்பதை எளிதாக அறிய உதவுகிறது. இதை சி மொழியில் **escape sequences** என்று அழைக்கிறார்கள். சி மொழி கொண்டே **Linux** எழுதப்பட்டுள்ளதால், இதுவும் அதே போல் செயல்படுகிறது.

bash$ echo hello\!

hello!

bash$ echo "hello\!"

hello\!

**bash$ echo **

>

bash$ echo "\\"

>

bash$ echo \a

a

bash$ echo "\a"

\a

bash$ echo x\ty

xty

bash$ echo "x\ty"

x\ty

bash$ echo -e x\ty

xty

bash$ echo -e "x\ty"

x **y**

கலைச்சொற்கள்:

உயர் நிலை கணினி மொழிகள் **- high level computer languages**

முனையக் குறுநிரல் **- shellscript**

முழுஎண் **- integer**

கோவை **- string**

மாறி **- variable**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் – 34
தப்பிடும் விசைகள் (escape sequences)

ஓடும் குறுநிரல் ஓடிடும் வழியில்
சிமொழி கொணர்ந்த சீருத விசையில்
தானே விலகி தீர்வுற் றோடிட
துணைசெய் விக்கும் தப்பிடும் விசையே.
- நிரற்பா 34

நிரற்பாவிளக்கம்:

தொடர்ந்து ஓடிக் கொண்டிருக்கும் குறுநிரலானது **(script),** அது தானாய் செயல்படும் வழியில் சிமொழியில் கொண்டுள்ள சீராய் உதவிடும் தப்புவிக்கும் விசைகளானவை **(escape sequence),** நிரலிலிருந்து தானாய் விலகி ஓடித் தீர்வைத் தேடி அடைந்திட உதவி செய்திடும் **.** பின்வரும் அட்டவணையானது எந்தெந்த விசைகளுக்கு என்னென்ன பொருள் என்பதை அறியத்தருகின்றது**.**

தப்பிடும் விசைகள்	விளக்கம்
\n	**means newline**
\r	**means return**
\t	**means tab**
\v	**means vertical tab**
\b	**means backspace**
\a	**means alert (beep or flash)**
\0xx	**translates to the octal ASCII equivalent of 0nn, where nn is a string of digits**

நிரல் **118**

```
#!/bin/bash
#niral118.sh
# escaped.sh: escaped characters
##############################################
##############
```

```
### First, let's show some basic escaped-character usage. ###
###############################################
##############
# Escaping a newline.
# ------------------
echo ""
echo "This will print
as two lines."
# This will print
# as two lines.
echo "This will print \
as one line."
# This will print as one line.
echo; echo
echo "============="
echo "\v\v\v\v"      # Prints \v\v\v\v literally.
# Use the -e option with 'echo' to print escaped characters.
echo "============="
echo "VERTICAL TABS"
echo -e "\v\v\v\v"   # Prints 4 vertical tabs.
echo "=============="
echo "QUOTATION MARK"
echo -e "\042"       # Prints " (quote, octal ASCII character 42).
echo "=============="

# The $'\X' construct makes the -e option unnecessary.

echo; echo "NEWLINE and (maybe) BEEP"
echo $'\n'           # Newline.
echo $'\a'           # Alert (beep).
                     # May only flash, not beep, depending on terminal.
```

```
# We have seen $'\nnn" string expansion, and now . . .

#
=================================================
===================== #
# Version 2 of Bash introduced the $'\nnn' string expansion construct.
#
=================================================
===================== #

echo "Introducing the \$\' ... \' string-expansion construct . . . "
echo ". . . featuring more quotation marks."

echo $'\t \042 \t'   # Quote (") framed by tabs.
# Note that  '\nnn' is an octal value.

# It also works with hexadecimal values, in an $'\xhhh' construct.
echo $'\t \x22 \t'  # Quote (") framed by tabs.
# Thank you, Greg Keraunen, for pointing this out.
# Earlier Bash versions allowed '\x022'.

echo

# Assigning ASCII characters to a variable.
# --------------------------------------
quote=$'\042'        # " assigned to a variable.
echo "$quote Quoted string $quote and this lies outside the quotes."

echo

# Concatenating ASCII chars in a variable.
triple_underline=$'\137\137\137'  # 137 is octal ASCII code for '_'.
```

```
echo "$triple_underline UNDERLINE $triple_underline"

echo

ABC=$'\101\102\103\010'          # 101, 102, 103 are octal A, B, C.
echo $ABC

echo
escape=$'\033'                   # 033 is octal for escape.
echo "\"escape\" echoes as $escape"
#                                no visible output.
echo
exit 0
```

[illegible]:

[illegible]

[illegible] 119:

```
#!/bin/bash
# niral119.sh
# Requires version 4.2+ of Bash.
key="no value yet"
while true; do
  clear
  echo "Bash Extra Keys Demo. Keys to try:"
  echo
  echo "* Insert, Delete, Home, End, Page_Up and Page_Down"
  echo "* The four arrow keys"
  echo "* Tab, enter, escape, and space key"
  echo "* The letter and number keys, etc."
  echo
  echo "    d = show date/time"
```

```
 echo "    q = quit"
 echo "================================="
 echo

# Convert the separate home-key to home-key_num_7:
if [ "$key" = $'\x1b\x4f\x48' ]; then
 key=$'\x1b\x5b\x31\x7e'
 #   Quoted string-expansion construct.
fi

# Convert the separate end-key to end-key_num_1.
if [ "$key" = $'\x1b\x4f\x46' ]; then
 key=$'\x1b\x5b\x34\x7e'
fi

case "$key" in
 $'\x1b\x5b\x32\x7e')  # Insert
  echo Insert Key
 ;;
 $'\x1b\x5b\x33\x7e')  # Delete
  echo Delete Key
 ;;
 $'\x1b\x5b\x31\x7e')  # Home_key_num_7
  echo Home Key
 ;;
 $'\x1b\x5b\x34\x7e')  # End_key_num_1
  echo End Key
 ;;
 $'\x1b\x5b\x35\x7e')  # Page_Up
  echo Page_Up
 ;;
 $'\x1b\x5b\x36\x7e')  # Page_Down
```

```
 echo Page_Down
;;
$'\x1b\x5b\x41')  # Up_arrow
 echo Up arrow
;;
$'\x1b\x5b\x42')  # Down_arrow
 echo Down arrow
;;
$'\x1b\x5b\x43')  # Right_arrow
 echo Right arrow
;;
$'\x1b\x5b\x44')  # Left_arrow
 echo Left arrow
;;
$'\x09')  # Tab
 echo Tab Key
;;
$'\x0a')  # Enter
 echo Enter Key
;;
$'\x1b')  # Escape
 echo Escape Key
;;
$'\x20')  # Space
 echo Space Key
;;
d)
 date
;;
q)
echo Time to quit...
echo
```

```
 exit 0
 ;;
 *)
  echo You pressed: \'"$key"\'
 ;;
esac
echo
echo "==================================="
unset K1 K2 K3
read -s -N1 -p "Press a key: "
K1="$REPLY"
read -s -N2 -t 0.001
K2="$REPLY"
read -s -N1 -t 0.001
K3="$REPLY"
key="$K1$K2$K3"

done
exit $?
```

நிரல் விளக்கம்:

இந்த நிரலானது பயனர் கொடுத்திடும் உள்ளிடு விசைகள் என்னென்ன என்பதை எடுத்துக் காட்டுவதாக அமைந்துள்ளது.

கலைச்சொற்கள்:

குறுநிரல் – **script**

சீருத விசையில் – **escape sequence**

தப்பிடும் விசை – **escape sequence**

துணைசெய் – **assist**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 35

வெளியேறும் வழிக்கட்டளை (exit command)

குறித்த செயலும் குறித்த வரிகளும்
தடையற நடந்து தெளிவுறு நிற்க
மேலும் கட்டளை மிகுந்தி ருப்பினும்
வெளியே செல்லும் வழிக்கட் டளையே.
- நிரற்பா 35

நிரற்பாவிளக்கம்:

குறிப்பிட்ட கட்டளைகளைக் கொண்டு குறிப்பிட்ட செயல்களானவை செய்து முடித்த பிறகு, இன்னும் வேறு ஏதேனும் கட்டளைகள் இருந்தால், அவற்றை இயக்காமல் நிரலிலிருந்து வெளியேற வழிக்கட்டளையானது உதவுகிறது.

நிரல் **120**

```
#!/bin/bash
#niral120.sh
echo hello
echo $?       # Exit status 0 returned because command executed successfully.

lskdf      # Unrecognized command.
echo $?      # Non-zero exit status returned -- command failed to execute.

echo

exit 113   # Will return 113 to shell.
        # To verify this, type "echo $?" after script terminates.
```

 # By convention, an 'exit 0' indicates success,

#+ while a non-zero exit value means an error or anomalous condition.

நிரல் விளக்கம்:

இங்கு **echo $?** என்ற கட்டளை, சுழியத்தை வெளியீடாகத் தருமாயின் அதற்கு முந்தைய கட்டளையானது பொறியில் சரியான முறையில் இயங்கியிருக்கிறது என்று பொருள். மாறாக, சுழியமல்லாத வேறு ஏதேனும் எண்ணை வெளியீடாகத் தருமாயின், முந்தைய கட்டளையானது தவறான முறையில் இயங்கியிருக்கிறது என்று பொருள். மேலே கொடுக்கப்பட்டுள்ள நிரல் இதனைச் செம்மையாக விளக்குவதாக அமைகிறது.

நிரல் **121:**

```
#!/bin/bash
# niral121.sh
true    # The "true" builtin.
echo "exit status of \"true\" = $?"     # 0

! true
echo "exit status of \"! true\" = $?"   # 1
# Note that the "!" needs a space between it and the command.
#    !true   leads to a "command not found" error
#
# The '!' operator prefixing a command invokes the Bash history
mechanism.

true
!true
# No error this time, but no negation either.
# It just repeats the previous command (true).


#
=================================================
============ #
```

```
# Preceding a _pipe_ with ! inverts the exit status returned.
ls | bogus_command     # bash: bogus_command: command not found
echo $?                # 127

! ls | bogus_command   # bash: bogus_command: command not found
echo $?                # 0
# Note that the ! does not change the execution of the pipe.
# Only the exit status changes.
#
=====================================================
============ #
```

நிரல் விளக்கம்:

இந்த நிரலும் ஏறத்தாழ அதே செய்தியினை விளக்குவதாக இருந்தாலும், இங்கு **true, !true** ஆகிய வரிகள் சேர்க்கப்பட்டுள்ளன. **true** என்பது உண்மையாக உள்ள போது இயங்குவதாகவும், **!true** என்பது உண்மையல்லாத எந்தவொரு கட்டளைக்கும் இயங்குவதாகவும் இருக்கும் வண்ணம் அமைக்கப்பட்டுள்ளது.

கீழே உள்ள அட்டவணையானது வெவ்வேறு வெளியேறும் நிலை எண்களையும் அவற்றிற்கான விளக்கங்களையும் அளிக்கிறது.

வெளியேறும் நிலை எண்	பொருள்	எடுத்துக்காட்டு
1	பொதுவான பிழைச்செய்திகள்	**let "var1 = 1/0"**
2	**Bash** ல் உள்ளவற்றைத் தவறாகப் பயன்படுத்தினால் வருவது.	**empty_function() {}**
126	குறிப்பிட்ட கட்டளை அழைக்கப்பட்டது ஆனால் இயக்கப்படவில்லை.	**/dev/null**
127	கட்டளைவரி இல்லை	**illegal_command**
128	தவறான அளவுருவுடன் வெளியேறுகிறது.	**exit 3.14159**
128+n	உறுதியான பிழைக்கான சமிக்ஞை அல்லது தாடை	**kill -9 $PPID of script**

130	[illegible] **ctrl+c** [illegible].	***Ctl-C***
255*	[illegible].	**exit -1**

Exit Status of a Script

- A script, like any other process, sets an exit status when it finishes executing. Shell scripts will finish in one of the following ways:

  Abort - If the script aborts due to an internal error, the exit status is that of the last command (the one that aborted the script).

  End - If the script runs to completion, the exit status is that of the last command in the script

  Exit - If the script encounters and exit command, the exit status is that set by that command

  - Syntax for the exit command is `exit (num)`. When the exit command is encountered the script ends right there and the exit status is set to `num`

71

[illegible]:

[illegible] **- commands**

[illegible] – **without interruption**

[illegible] – **more commands need to run**

[illegible] – **exit command**

[illegible] – **zero**

[illegible] – **non-zero**

[illegible] – **alternatively**

[illegible] – **signal**

[illegible] – **parameter**

உறுதியான பிழை – fatal error

(கற்போம்)

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -225- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 36
உள்ளார்ந்த கட்டமைப்புக் கட்டளைகள் (Internal Commands and Builtins)

முன்னிரு உள்ளார் மூத்த கட்டளை
என்னென வகையில் ஏதுகிறது என்பதை
பின்வரும் நிரல்கள் பகுத்தாய் ஆய்ந்து
வெளியிடு காட்டி விளக்கு கிறதே.
- நிரற்பா 36

நிரற்பாவிளக்கம்:

பின்வரும் குறுநிரல்களானவை ஏற்கனவே பொறியில் உள்ள பல்வேறு முன்னிருப்பு கட்டளைகளைக் கொண்டு என்னென்ன செய்யல்லாம் என்பதை எடுத்துக்காட்டி விளக்குகின்றன.

நிரல் **122**

```
#!/bin/bash
#niral122.sh
#Internal Commands and Builtins
#A script that spawns multiple instances of itself
# spawn.sh

PIDS=$(pidof sh $0)   # Process IDs of the various instances of this script.
P_array=( $PIDS )    # Put them in an array (why?).
echo $PIDS           # Show process IDs of parent and child processes.
let "instances = ${#P_array[*]} - 1"  # Count elements, less 1.
                                      # Why subtract 1?
echo "$instances instance(s) of this script running."
echo "[Hit Ctl-C to exit.]"; echo
```

```
sleep 1             # Wait.
sh $0               # Play it again, Prasanna.

exit 0              # Not necessary; script will never get to here.
                    # Why not?

#  After exiting with a Ctl-C,
#+ do all the spawned instances of the script die?
#  If so, why?

# Note:
# ----
# Be careful not to run this script too long.
# It will eventually eat up too many system resources.

#  Is having a script spawn multiple instances of itself
#+ an advisable scripting technique.
#Generally, a Bash builtin does not fork a subprocess when it
executes within a script. An external system command or filter in a
script usually will fork a subprocess.
```

[illegible]:

[illegible], [illegible]. Ctrl+c [illegible]. [illegible].

[illegible] 123:

```
#!/bin/bash
# niral123.sh

# Embedding a linefeed?
```

```
echo "Why doesn't this string \n split on two lines?"
# Doesn't split.

# Let's try something else.
echo

echo $"A line of text containing
a linefeed."
# Prints as two distinct lines (embedded linefeed).
# But, is the "$" variable prefix really necessary?

echo
echo "This string splits
on two lines."
# No, the "$" is not needed.
echo
echo "---------------"
echo

echo -n $"Another line of text containing
a linefeed."
# Prints as two distinct lines (embedded linefeed).
# Even the -n option fails to suppress the linefeed here.

echo
echo
echo "---------------"
echo
echo
# However, the following doesn't work as expected.
# Why not? Hint: Assignment to a variable.
string1=$"Yet another line of text containing
```

```
a linefeed (maybe)."

echo $string1
# Yet another line of text containing a linefeed (maybe).
#                                    ^
# Linefeed becomes a space.
```

[illegible]:

[illegible]

[illegible] 124:

```
#!/bin/bash
#niral124.sh
# printf demo

declare -r PI=3.14159265358979        # Read-only variable, i.e., a constant.
declare -r DecimalConstant=31373

Message1="Greetings,"
Message2="Earthling."

echo

printf "Pi to 2 decimal places = %1.2f" $PI
echo
printf "Pi to 9 decimal places = %1.9f" $PI   # It even rounds off correctly.

printf "\n"                                  # Prints a line feed,
                                             # Equivalent to 'echo' . . .
```

```
printf "Constant = \t%d\n" $DecimalConstant  # Inserts tab (\t).

printf "%s %s \n" $Message1 $Message2

echo

# ==========================================#
# Simulation of C function, sprintf().
# Loading a variable with a formatted string.

echo

Pi12=$(printf "%1.12f" $PI)
echo "Pi to 12 decimal places = $Pi12"      # Roundoff error!

Msg=`printf "%s %s \n" $Message1 $Message2`
echo $Msg; echo $Msg

#  As it happens, the 'sprintf' function can now be accessed
#+ as a loadable module to Bash,
#+ but this is not portable.

exit 0
```

Formatting error messages is a useful application of printf

```
E_BADDIR=85

var=nonexistent_directory

error()
{
```

```
  printf "$@" >&2
  # Formats positional params passed, and sends them to stderr.
  echo
  exit $E_BADDIR
}

cd $var || error $"Can't cd to %s." "$var"
```

நிரல் விளக்கம்: **(printf command in shell script)**

இலினக்சு இயங்குதளம் முழுக்கவே சிமொழியில் இழைத்து எழுதப்பட்டமையால், சி மொழியில் பயின்று வரும் கட்டளைகளை நாம் மேம்போக்காக சிற்சில மாற்றங்களுடன் எழுதி வெளியீட்டினை அறியலாம். **printf** கட்டளையானது சி மொழியில் உள்ள ஒரு பதிப்பிக்கும் கட்டளைவரியாகும், நாம் இங்கும் அதைக் கொண்டே உரைகளைக் கையாண்டு விடையறியலாம், அதில் பயின்று வரும் தப்பிடும் விசைகளையும் இதில் பயன்படுத்தலாம்.

கலைச்சொற்கள்:

முன்னிரு - **default**

ஏதுகிறது - காரணமாகிறது - **reason of**

நிரல்கள் - **scripts**

வெளியிடு - **output**

உட்பொதிய - **embedded**

உரை - **text**

உள்ளார் - **built-in**

தப்பிடும் விசைகள் - **escape sequences**

(கற்போம்)

செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் - 37

இரட்டை அடைப்புக் குறி

கணிதச் சோதனை கணினியில் செய்திட
சிறிய நிரலில் செம்மை பொருந்த
சீர்மிகு வண்ணம் செயற்பட வந்திடும்
இரட்டை அடைப்பு ஏரணக் குறியே.
நிரற்பா - 37

நிரற்பா விளக்கம்:

கணினிச் சோதனைகளை எளிமையான வண்ணம் கணினியில் செய்திட இரட்டை அடைப்புக்குறியானது ஏரணப்பிழையற பயன்படுத்திடல் வேண்டும் .

நிரல் **124:**

```
#!/bin/bash
# arith-tests.sh
# Arithmetic tests.

# The (( ... )) construct evaluates and tests numerical expressions.
# Exit status opposite from [ ... ] construct!

(( 0 ))
echo "Exit status of \"(( 0 ))\" is $?."         # 1

(( 1 ))
echo "Exit status of \"(( 1 ))\" is $?."         # 0

(( 5 > 4 ))                                      # true
echo "Exit status of \"(( 5 > 4 ))\" is $?."     # 0
```

```
(( 5 > 9 ))                                      # false
echo "Exit status of \"(( 5 > 9 ))\" is $?."     # 1

(( 5 == 5 ))                                     # true
echo "Exit status of \"(( 5 == 5 ))\" is $?."    # 0
# (( 5 = 5 ))  gives an error message.

(( 5 - 5 ))                                      # 0
echo "Exit status of \"(( 5 - 5 ))\" is $?."     # 1

(( 5 / 4 ))                                      # Division o.k.
echo "Exit status of \"(( 5 / 4 ))\" is $?."     # 0

(( 1 / 2 ))                                      # Division result < 1.
echo "Exit status of \"(( 1 / 2 ))\" is $?."     # Rounded off to 0.
                                                 # 1

(( 1 / 0 )) 2>/dev/null                          # Illegal division by 0.
#           ^^^^^^^^^^^
echo "Exit status of \"(( 1 / 0 ))\" is $?."     # 1

# What effect does the "2>/dev/null" have?
# What would happen if it were removed?
# Try removing it, then rerunning the script.

# ======================================= #

# (( ... )) also useful in an if-then test.

var1=5
var2=4
```

```
if (( var1 > var2 ))
then #^      ^      Note: Not $var1, $var2. Why?
  echo "$var1 is greater than $var2"
fi     # 5 is greater than 4

exit 0
```

நிரல் விளக்கம்:

மேற்கூறிய நிரலில், இரண்டு மாறிகளை வெவ்வேறு விதமாக ஒப்பீடு செய்வது என்பதைப் பற்றி விளக்கப்பட்டுள்ளது. ஒப்பீடு என்பது, இரண்டு எண்கள் கீழ்வருமாறு எந்த வகையிலும் இருக்கலாம்.

1. நிகராக இருப்பது (equals)
2. அதிகமாக இருப்பது (more than)
3. குறைவாக இருப்பது (less than)
4. அதிகமாக நிகராக இருப்பது (more than or equal)
5. குறைவாக நிகராக இருப்பது (less than or equal)

ஆகியவை கையாளப்பட்டுள்ளன.

நிரல் 125:

```
#!/bin/bash
# broken-link.sh
# Written by PNA Prasanna
# Used in ABS Guide with permission.

#  A pure shell script to find dead symlinks and output them quoted
#+ so they can be fed to xargs and dealt with :)
#+ eg. sh broken-link.sh /somedir /someotherdir|xargs rm
#
#  This, however, is a better method:
#
#  find "somedir" -type l -print0|\
#  xargs -r0 file|\
#  grep "broken symbolic"|
```

```
#  sed -e 's/^\|: *broken symbolic.*$/"/g'
#
#+ but that wouldn't be pure Bash, now would it.
#  Caution: beware the /proc file system and any circular links!
##############################################
################

#  If no args are passed to the script set directories-to-search
#+ to current directory.  Otherwise set the directories-to-search
#+ to the args passed.
#####################

[ $# -eq 0 ] && directorys=`pwd` || directorys=$@


#  Setup the function linkchk to check the directory it is passed
#+ for files that are links and don't exist, then print them quoted.
#  If one of the elements in the directory is a subdirectory then
#+ send that subdirectory to the linkcheck function.
##########

linkchk () {
    for element in $1/*; do
      [ -h "$element" -a ! -e "$element" ] && echo \"$element\"
      [ -d "$element" ] && linkchk $element
    # Of course, '-h' tests for symbolic link, '-d' for directory.
    done
}

#  Send each arg that was passed to the script to the linkchk()
function
```

```
#+ if it is a valid directoy.  If not, then print the error message
#+ and usage info.
##################
for directory in $directorys; do
   if [ -d $directory ]
    then linkchk $directory
    else
       echo "$directory is not a directory"
       echo "Usage: $0 dir1 dir2 ..."
   fi
done

exit $?
```

நிரல் விளக்கம்:

இந்த நிரலில் எண்கள் மட்டும் கையாளப்படாமல், கோப்புகள் மற்றும் அடைவுகள் கையாளப்பட்டுள்ளன. அவை சரிவர ஒன்று மற்றொன்றுடன் சேர்ந்து ஒப்பிடப்படும் வண்ணம் கையாளப்பட்டுள்ளன.

கலைச்சொற்கள்:

கணிதச் சோதனை – **arithmetic operations**

இரட்டை அடைப்புக் குறி – **double brackets**

ஏரணம் - **logic**

(கற்போம்)

முடிவுரை:

முடிவுரை என்றதும் இவ்வளவுதான் என்று எண்ண வேண்டாம். இந்நூலானது
ஒரு எளிய கற்றலுக்கான இனிய தொடக்கம் தான். தமிழில் கற்றோருக்கு எளிமையாகப் புரியும் படி அளிக்கப்பட்டுள்ளது. நூலில் முற்றும் போடாமல், “கற்போம்” என்று வரைந்திருப்பதால், தொடர்ந்து ஷெல் ஸ்கிரிப்ட்டினைக் கற்று வர இந்நூல் உதவும் என்று நம்புகிறேன்.
ஒரு புதிய பெரிய நிரலை எங்கேனும் கண்டால், அதை இந்நூலைக் கொண்டு, ஒப்பிட்டு ஒரு குறிப்பாக பார்த்தால் அந்தப்புதிய நிரலின் போக்குப் புரியும் வண்ணமே இந்நூல் எடுத்தாளப்பட்டுள்ளது.

எளிய வகையில் இனிய தமிழில்
பிழையற இயற்றிப் பழக‌
கடின நிரலும் கையில் தவழும்
“கற்போம்” கொண்டு கற்று வரவே!!
- நிரற்பா 38

செம்மொழியில் கற்போம்ஷெல் ஸ்கிரிப்ட் -237- பெங்களூரு பணியா. பிரசன்னா எம்.எஸ்.சி. எம்.ஃபில்.,

நூல் ஆசிரியர் :

நான்...

பெங்களூரு. பணியா. பிரசன்னா எம்.எஸ்.சி . எம் .பில் .

திண்டுக்கல் தூய மரியன்னைப் பள்ளியில் முடித்து, திருச்சி தூய வளனார் கல்லூரியில் கணினி அறிவியலில் முதுகலைப் பட்டம் படித்து முடித்துள்ளேன் . மதுரை காமராசர் பல்கலைக்கழகத்தில் எம் .பில் தொடர்ந்துள்ளேன். இன்று பெங்களூரில் ஓர் மென்பொருள் நிறுவனத்தில் மென்பொருள் கட்டுமானர் ஆகவேலை பார்த்துக்கொண்டிருக்கிறேன் .

விர்ஜின் பிரசன்னாவை ***2008*** இல் திருமணம் முடித்தவன், இன்று ஹேமில்டன், ஹேரிங்டன் ஆகிய இரு ஆண் மகவுகளின் தந்தை . நேரம் கிடைக்கும் பொழுதெல்லாம், பொழுதாக்கமாக கணினிக்கதைகள், கட்டுரைகள் எழுதுவது, கணினித் தொடர் எழுதுவது ஆகியவற்றைச் செய்துஇன்புறுகிறேன். தமிழ் கம்ப்யூட்டர் இதழில் வெளியான செம்மொழியில் கற்போம் ஷெல் ஸ்கிரிப்ட் என்ற தொடரைத் தொகுத்து இந்நூலில் வழங்கியுள்ளேன்.

பால் ஜெயசீலன் நிர்மல் ஆரோக்கிய பிரசன்னா என்ற இயற்பெயர் சுருங்கி, பா.நி .ஆ. பிரசன்னா என்று சான்றிதழ்களுக்காக மாற்றப்பட்டது. அதுவே மேலும் சுருங்கி பணியா. பிரசன்னா என்றாயிற்று.

நன்றி...